

FINAL FY04 REPORT FOR

“Naval Automation and Information Management
Technology”

(N00014-04-1-0507)

Jerry Pratt, Peter Neuhaus, Jeffrey M. Bradshaw, Niranjan Suri, James Allen, Lucian Galescu

Florida Institute for Human and Machine Cognition (IHMC), Pensacola, FL

Submitted to:

Gary Toth
Office of Naval Research
800 N. Quincy Street
Arlington VA 22217-5660



INSTITUTE FOR HUMAN & MACHINE COGNITION

University of West Florida, 40 South Alcaniz Street, Pensacola, FL 32502

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 03-01-2006		2. REPORT TYPE Final Technical Report		3. DATES COVERED (From - To) 01-05-2004 to 31-08-2005	
4. TITLE AND SUBTITLE Naval Automation and Information Management Technology				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER N00014-04-1-0507	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Dr. Jerry Pratt, Dr. Jeffrey M. Bradshaw, Dr. James Allen, Dr. Lucian Galescu, Niranjani Suri				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Florida Institute for Human and Machine Cognition 40 S. Alcaniz St. Pensacola FL 32502				8. PERFORMING ORGANIZATION REPORT NUMBER Final	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Ballston Centre Tower One 800 N. Quincy St. Arlington VA 22217				10. SPONSOR/MONITOR'S ACRONYM(S) ONR	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Unrestricted <div style="text-align: center;"> DISTRIBUTION STATEMENT A Approved for Public Release Distribution Unlimited </div>					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <p>In future military scenarios, large numbers of unmanned ground, air, underwater, and surface vehicles will work together, coordinated by an ever smaller number of human operators. In order to be operationally efficient, effective and useful, these robots must have competent physical and sensing abilities, must be able to perform complex tasks semi-autonomously, must be able to coordinate with each other, and must ultimately be observable and controllable in a useful and intuitive fashion by human operators. In addition, future soldiers will be outfitted with exoskeletons to enhance their capabilities, whether carrying heavy loads, swimming, carrying heavy loads, or scaling walls.</p> <p>Under the Naval Automation and Information Management Technology Program (NAIMT), The Institute for Human and Machine Cognition (IHMC) of the University of West Florida has conducted advanced research on unmanned systems and exoskeletons in the areas of (1) underwater exoskeletons for enhancing speed and endurance of Navy divers, (2) human-agent teamwork and agile computing and (3) mixed initiative human control. Progress made in FY04 in each of these three areas is described below.</p>					
15. SUBJECT TERMS Artificial Intelligence, Human-Centered Computing, Augmented Cognition, Biologically-Inspired Robots, Human-Agent Teamwork, Mixed-Initiative Human Control, Agile Computing					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 44	19a. NAME OF RESPONSIBLE PERSON Jerry Pratt
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) 850-202-4481

I. Summary

In future military scenarios, large numbers of unmanned ground, air, underwater, and surface vehicles will work together, coordinated by an ever smaller number of human operators. In order to be operationally efficient, effective and useful, these robots must have competent physical and sensing abilities, must be able to perform complex tasks semi-autonomously, must be able to coordinate with each other, and must ultimately be observable and controllable in a useful and intuitive fashion by human operators. In addition, future soldiers will be outfitted with exoskeletons to enhance their capabilities, whether carrying heavy loads, swimming, carrying heavy loads, or scaling walls.

Under the Naval Automation and Information Management Technology Program (NAIMT), The Institute for Human and Machine Cognition (IHMC) of the University of West Florida has conducted advanced research on unmanned systems and exoskeletons in the areas of (1) underwater exoskeletons for enhancing speed and endurance of Navy divers, (2) human-agent teamwork and agile computing and (3) mixed initiative human control. Progress made in FY04 in each of these three areas is described below.

II. Performance Improving Self Contained Exoskeleton for Swimming (PISCES)

The ability for Navy divers to swim faster and farther while using less energy would enhance their capabilities, enabling them to accomplish missions previously not possible. Current technology to propel divers underwater is either large and bulky, requires the use of the hands, or restricts maneuverability. We have developed a Performance Improving Self Contained Exoskeleton for Swimming, PISCES, which promises to augment a diver's natural capabilities.

The PISCES Exoskeleton (see Fig. 1) is an underwater, robotic, human-power augmentation device. This device is designed to help a diver swim faster and farther by means of an intuitive interface that amplifies the user's motions. PISCES has all the control needs on board will eventually be powered by on board batteries. The connection to the user is a rigid torso brace that transmits the propulsion force from the device to the user. The exoskeleton frame is connected to the user's legs at the thigh and the shank. Each of these four connection points consists of a compliant force measuring device to measure the torque (force) the user is exerting about the two hip joints and the two knee joints. The force measured at each of these points is used to drive the four motors (one at each hip and one at each knee).

The motors are DC brushless motors with hall sensors and encoders. The output of the motor is connected to a harmonic gearbox speed reducer. The motor, halls, and encoder for each actuator are connected via a single cable to the electronics box. An encoder measures the deflection of a spring in determining the user's force. These encoders each have a separate cable that connects to the electronics box. The electronics box houses the

microprocessor, the motor amplifiers, and all the other miscellaneous electronics. The exoskeleton is currently powered by a tether; we plan to use batteries in separate containers and connected to the electronics box through a cable.

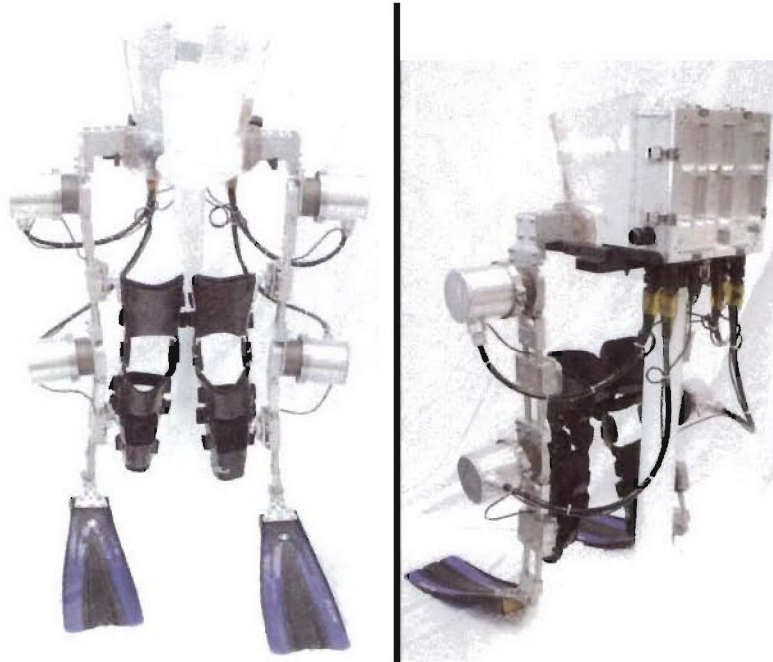


Figure 1: Front and back view of the PISCES exoskeleton on a display stand (shown without user). The user straps the clear plastic brace at the top around the torso and the black knee braces around the thigh and shank of each leg. Two sensors at each leg between the brace and exoskeleton are used to command the motors at the hip and knee joints. The electronics system is contained in a waterproof box the user wears on his or her back.

PISCES Design

Goals

The design goals of PISCES are:

- Enhanced endurance – The user should be able to swim at 1.0m/s for up to four hours without exhaustion. In comparison, experienced combat divers can swim for up to four hours at a speed of 0.5m/s.
- Increased burst speed – The user should be able to sprint at a speed of 1.5m/s for up to 30 minutes. In comparison, experienced swimmers can sustain that speed for a couple of minutes [1].
- Natural interface – The exoskeleton should be operated naturally. In order to swim, the user should simply kick. In order to speed up, the user should kick faster. A well designed device will directly amplify, without impeding, the user's natural motions so that it truly is an extension of the wearer's body.

- Low impedance – The device should not impede the natural kick motions of the user. The interactive forces between it and the user should be minimal.

Design

In this section we discuss the design and hardware of the PISCES exoskeleton. Where applicable we provide manufacturer and part number information of the hardware used.

Layout of actuators and degrees of freedom

For a human swimming underwater the most efficient muscle power to use for propulsion is the leg muscles with the flutter kick as the most popular kick style. This kick profile involves motion of the hip, knee, and ankle joints. However, from studying human swimming, the role of the ankle joint seems to be mainly passive; during the power stroke (down), the ankle is pushed to the joint limit, and during the return (up), the ankle is positioned to minimize the drag on the foot. In addition, the motion is in the sagittal plane. Therefore, the targeted joints to actuate are the hip and knee joint in flexion and extension, with motion in the sagittal plane. The result is a total of four actuated degrees of freedom (Figure 2)

The PISCES exoskeleton structure parallels the legs, residing distal to the sagittal plane. It is firmly connected to the torso of the user by means of a custom molded polycarbonate torso brace (Figure 1). This brace transfers the propulsive thrust from the exoskeleton to the user and is the only rigid connection point between the user and the exoskeleton.

There are two passive joints on the exoskeleton frame (Figure 2). A passive joint at the hip allows for abduction and adduction. The second passive joint lies just below the knee joint, with its axis similarly positioned to the passive joint at the hip. This additional passive degree of freedom is required to allow the user to bend the torso in the frontal plane.

User interface

A key requirement for a successful exoskeleton is that it has a natural and intuitive interface. Ideally the user can put the device on and start using it without training. In order to accomplish this, there must be a means built into the device for detecting the user's intent.

In fin swimming, the intent of the user is conveyed through the motion of his or her legs. During forward propulsion the majority of motion occurs in the sagittal plane. Therefore, the PISCES exoskeleton measures intent through connections at the user's legs at the thigh and shank (Figure 3). At each of these connections, a custom device measures the relative force between the user's leg and the exoskeleton (Figure 4). The device consists of a linear slide whose axis is in the direction of flexion and extension motion of the leg. The linear slide is spring loaded so that the force between the exoskeleton and the user is proportional to their relative displacement.

The user wears a knee brace that has two balls, which is part of a ball joint, mounted at the thigh and shank sections. The user first puts on the knee brace, then snaps the balls

into the quick release sockets that are mounted on the carriage of the linear slides. The ball joint accommodates slight misalignment and physical variations among the different users. The quick release enables easy donning and doffing of the exoskeleton.

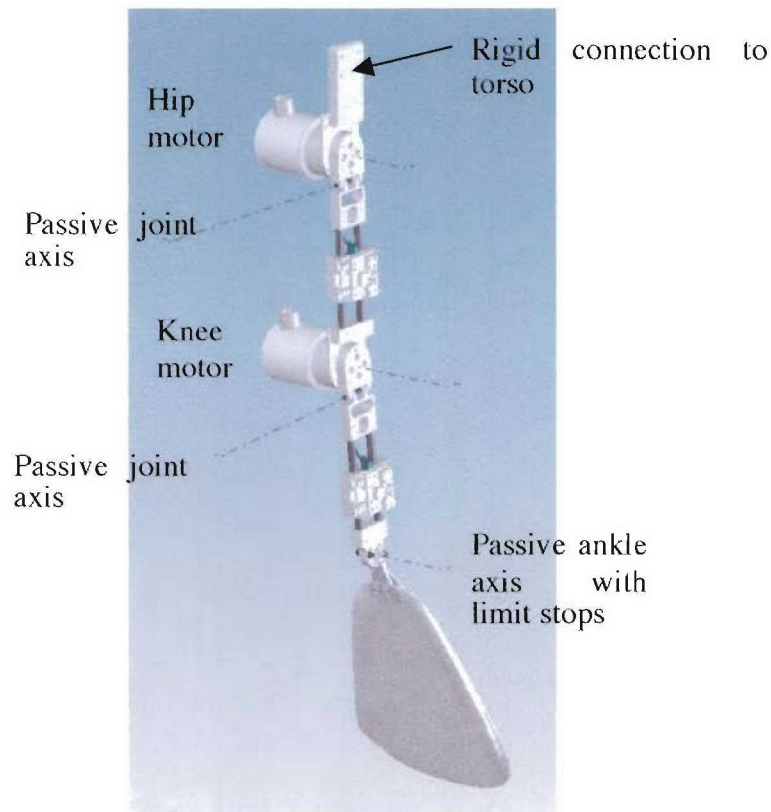


Figure 2: CAD figure of exoskeleton right leg showing powered and passive axes. There are two powered axes, one at the hip and one at the knee. There are three passive axes, one below the hip, one below the knee, and one at the end effector.

The linear position of the carriage block relative to the base drives a rack and pinion which turns a rotary encoder. The carriage has springs connecting it to the base, so that the distance the carriage moves from the center position is proportional to the force applied to the carriage. The value of the relative force can be used as an input into the control algorithm which drives the servomotors.



Figure 3: Picture of knee brace and attachment to exoskeleton frame. There are two attachment points between the brace and the exoskeleton frame, one at the thigh, and one at the shank. At each point, there is a quick connect ball joint for easy donning and doffing.

Power system

The heart of the PISCES actuation is four brushless DC motors. Electric motors offer several advantages over other types of actuators. They are easy to control, have good power to weight ratios, good power to volume ratios, are widely available, and can be waterproofed relatively easily. For initial tests, direct current electrical power is supplied by a DC power supply on dry land. The system voltage is limited to 30VDC for safety reasons because voltages higher than this pose risk of electrocution for the user. From a power transmission perspective, it is desired to use the highest practical and safe voltage to minimize current, and minimize wire size. Initial tests will continue to use power supplied through an umbilical. Once an estimate of the power used by PISCES is determined, a battery pack system will be designed for untethered operation.

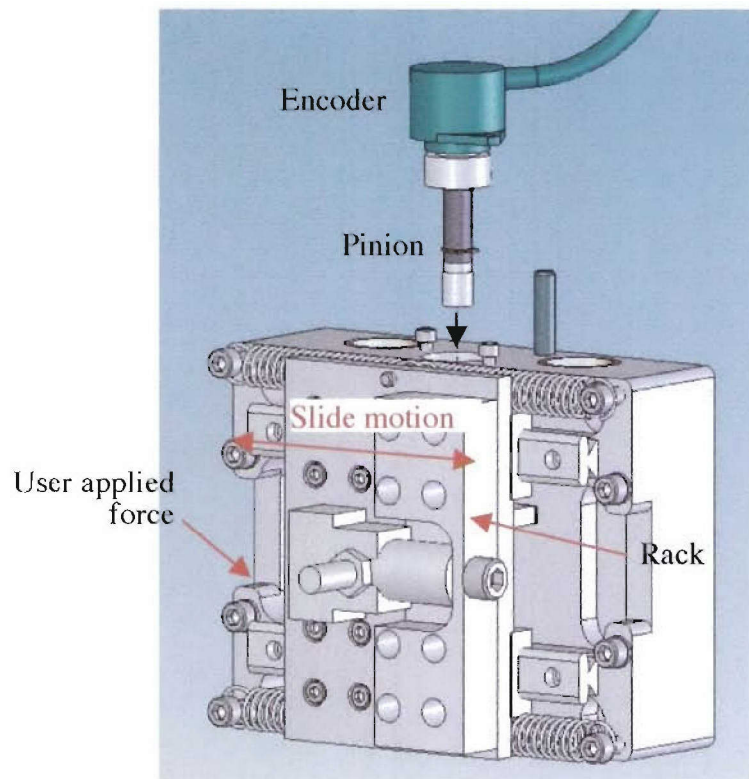


Figure 4: Exploded view of force measurement system. The encoder assembly has been moved up to reveal the pinion. The user applies force to the carriage parallel to the linear rails. Motion of the slide moves the rack and turns the pinion.

Actuator design

The actuator requirements for PISCES were determined from speed and power estimates for human underwater fin swimming. The maximum kick frequency of the average person fin swimming at maximum speed is 62 kicks per minute [2]. A higher angular velocity occurs at the knee joint because it oscillates with a larger motion than the hip joint. Using a sinusoidal joint position profile having amplitude of 110 degrees, the resulting peak angular velocity is 60 rpm. An initial power estimate of 500 Watts total was used to determine the approximate power rating for the motor. This estimate is based on active drag measurements of humans swimming [3][4]. Assuming that each leg must output the full total power, and that each motor supplies half the power for that leg, a minimum continuous power rating of 250 Watts was specified for each motor.

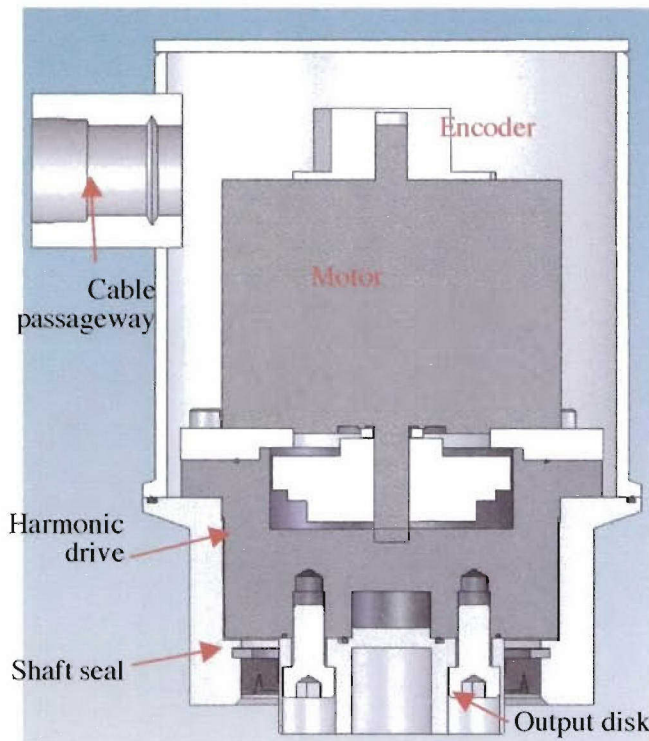


Figure 5: Cross sectional view of an actuator. The motor and gearbox are contained in a waterproof housing. The cable enters the housing at the upper left through a liquid tight strain relief and is potted on the inside with epoxy. The output disk is mounted to the output of the harmonic drive and rotates within the shaft seal.

Each of the four actuators consists of a motor, gear reducer, and feedback device all contained in a waterproof housing (see Figure 5). The motors selected for the exoskeleton are Moog/Litton brushless motors, part number BN43-25EU-01. This motor has a continuous power rating of 328 watts, and a terminal voltage of 24 VDC. A rotary encoder from Agilent (part number HEDL-5640#A13) was mounted onto the back of the motor to provide incremental position feedback. A harmonic drive from HD Systems (part number CSG-250-100-2UH-SPA-1733) with a gear ratio of 100:1 was used for speed reduction. Mounted to the output of the harmonic drive is a machined aluminum disk which provides the rotating interface that is exposed to water. The outer surface of this disk rotates and contacts the fixed surface of a shaft seal (from JM Clipper, part number JM 0200 152155066HP). Each actuator is connected to the control box via a single cable. This cable carries motor coil current, motor hall signal and power, and encoder signal and power. Because of the environment, wire gage size, number of conductors, and noise isolation requirements, this cable was custom ordered from Falmat, Inc. The point of entry of the cable into the housing is sealed with a liquid tight strain relief as well as potted on the inside with epoxy. The two halves of the actuator housing are sealed with an o-ring and connected with a v-band style clamp.

Control Strategy

The preliminary control strategy we implemented on PISCES was to convert the user's intent, as measured by the force measurement systems, into commands to the motors.

Each servomotor used the same control algorithm. The input to the control algorithm for the hip motor was the reading from the thigh force measurement device, and knee joint motor was controlled by the shank force measurement. The control strategy was:

$$\omega_{des} = k_{fv} F_{app} \quad (1)$$

where,

ω_{des} is the desired motor velocity

k_{fv} is the force to velocity gain

F_{app} is the user applied force

The force to velocity gain controlled the effective force amplification of the motor. The higher the number, the more force amplification or muscle augmentation from the exoskeleton. The desired velocity was then used in a simple proportional feedback controller:

$$u = k_p (\omega_{des} - \omega_{act}) \quad (2)$$

where,

u is the input torque to the motor

k_p is the proportional feedback gain

ω_{act} is the actual motor velocity

Electronics System

The electronics system is self-contained inside a custom designed waterproof box (see Figure 6). It consists of a microcontroller, input and output boards, motor amplifiers, and control circuitry.

The controller used is the CoreModule 600 from Ampro, which is a 400MHz low voltage Celeron PC104 embedded controller. The operating system is real time Linux, from TimeSys, Inc. The interface with the eight encoders is through a quadrature counter 4I36 board from Mesa Electronics. The motor amplifiers used are Accelnet Module part number ACM-180-20, from Copley Controls Corp. The input to the amplifiers comes from a PWM board from Real Time Devices (part number DM6816HR). For safety, a magnet with a pull cord must be placed in a special recess of the outside of the box to enable the motor amplifiers. When the operator needs to disable the motors, he or she pulls a cord to remove the magnet which is then detected by a reed switch on the interior of the box.

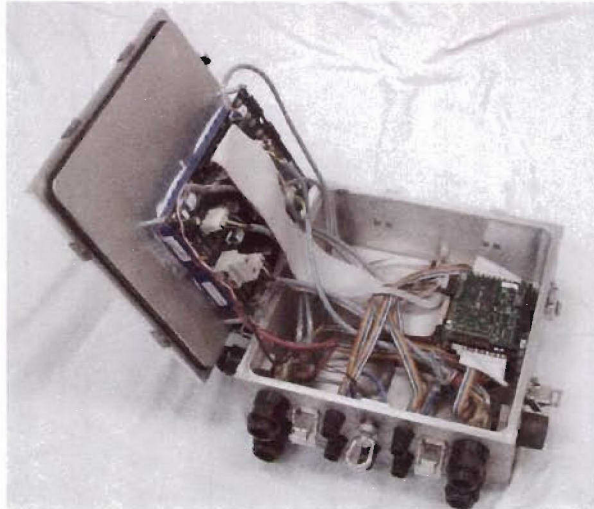


Figure 6: Picture of electronics box with cover opened. This box features a double o-ring seal and waterproof electrical connectors.

There are ten electrical connectors on the control box. Four of these are connectors for the actuators (Impulse part number XSL-12), four are connectors for the force measurement encoders (Impulse part number XSJ-9), one is a connector for power (Impulse part number XSL-4), and one is a connector for Ethernet communication (Impulse part number XSJ-9).

The control code for the exoskeleton is written in real time Java. During the development and testing of the exoskeleton, communication with the controller is made through the Ethernet connection. Through this connection operational data is acquired and control parameters can be modified in real time.

End effector

At the end of the shank link of the exoskeleton is a single degree of freedom passive pin joint which provides extension and flexion of the end effector, or exoskeleton fin (see Figure 7). This joint can be utilized on two different modes. In one mode, the joint limit hard stops are positioned to freely allow the foot to move between the two limits. These limits can be positioned in any of the holes for the desired range of motion. In the other mode, the exoskeleton ankle is pinned to the ankle by means of a screw, preventing any movement of the ankle. The preferred mode and settings will be determined through testing.

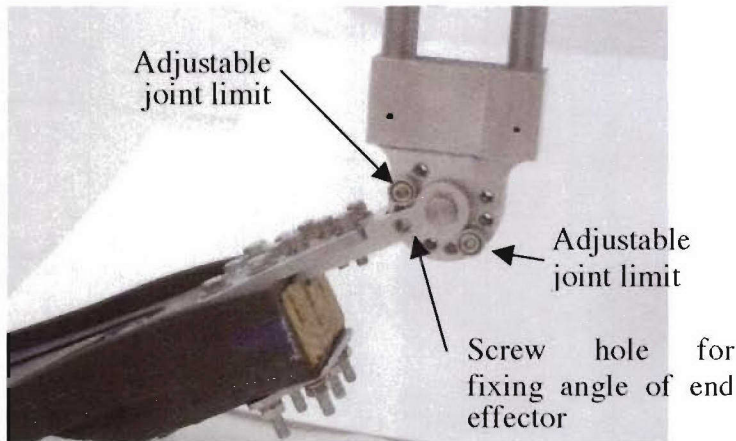


Figure 7: Picture of end effector joint. The joint is configured to move passively between the two joint limit stops. The joint can alternatively be configured for a fixed angle by placing a screw through the denoted hole.

Testing

Passive joint testing and range of motion

One of the important requirements of an exoskeleton is that it assists, and does not hinder, the natural movements of the user. For the PISCES exoskeleton, this meant that the user should be able swim naturally while wearing the unpowered device. To test this, the motors were replaced with passive joints, and the linear slides were locked in position (i.e. not allowed to move). The user put on the exoskeleton and tried to swim underwater. This passive joint test of the device was performed three separate times. After each test, an analysis was performed of the design, examining hardware failures and limits on the user's range of motion. Redesign and modification of the hardware was made following the analysis. For the final version, prior to installing the motors, the user was able to swim naturally with a minimal restriction of the range of motion.

Force tracking

During operation, the exoskeleton's motors are moved based on the movements of the user. As the user moves his or her legs, the applied force at each link used to drive the respective motor is derived based on the closed loop control algorithm defined in (1) and (2). For the initial force tracking test the output of the motor was decoupled from the force measurement assembly so that the movement of the motor did not affect the applied force.

The force was manually applied at two different frequencies, 0.5 Hz and 1Hz, with the latter simulating the maximum kick rate of a human. In examining the closed loop response of the unloaded motor (see Figure 8), one will note that the actual and desired curves have the same shape, but there is an attenuation in the magnitude. This is to be expected; because of the frictional losses in the system, an error in tracking is required in order to have a non-zero motor velocity (2). The magnitude of this error could be reduced with a feed forward term based on empirical measurements of the Coulomb and viscous

friction. However, the actual magnitude of ω_{des} is controlled by k_f , which is in effect the exoskeleton augmentation factor. Therefore, to have the actual motor velocity track the absolute magnitude of ω_{des} is not critical. From a user's point of view, it is more important that the phase response be good, and that the magnitude response be repeatable and constant with respect to frequency. In Figure 8b) we see a phase lag of about 30 degrees in the output at an input of 1Hz, which is the minimum expected input frequency.

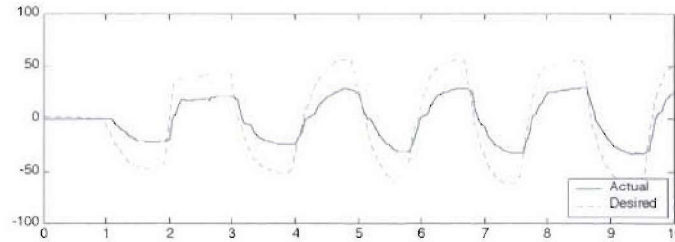


Figure 8: Closed loop motor tracking with no load at a) Approximately 0.5Hz desired signal. Note that there is little phase error. b) Approximately 1 Hz. Note that the phase error is about 30 degrees. For both plots, the attenuation is the same.

Discussion

PISCES, an underwater swimming exoskeleton was designed, built, and preliminarily tested. The robotic device is designed to increase the speed and endurance of a human swimming underwater. The exoskeleton has been tested in the water for fit, comfort, and range of motion. The force measurement system, the electronic system, and the actuators have been tested underwater for an extended period of operation.

Future work will include testing PISCES with a user in the water to optimize the control parameters and the hardware adjustments. Through testing, the control parameter k_f will be tuned for optimal performance. The preferred mode and limits for the exoskeleton ankle joint will also be determined from underwater testing with a user. Once the software and hardware parameters have been determined, PISCES will be ready for performance evaluation. This will include determining the metabolic benefit through measurements of oxygen consumption of the user during swimming with and without the exoskeleton.

PISCES was designed to augment the fin swimming flutter kick. Future versions could be designed for other styles of swimming, such as the dolphin kick. With the appropriate interface such a device could make the user not only swimming as well as a dolphin, but feel as though they were a dolphin. The ultimate goal for this project is to combine the swimming and terrestrial exoskeleton into a single robotic device. This amphibious exoskeleton would enable the user to swim fast and far, get to shore, and then walk a long distance while carrying a heavy load.

Resulting Papers

Peter D. Neuhaus, Michael O'Sullivan, David Eaton, John Carff, and Jerry E. Pratt, 2004. Concept Designs for Underwater Swimming Exoskeletons. *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, 4893-4898.

References

- [1] J. Hardy, "2001 testers' choice fins," *Scuba Diving*, Nov/Dec 2000.
- [2] D.R. Pendergast, J. Mollendorf, C. Logue, and S. Samimy, "Evaluation of fins used in underwater swimming," *J. of the Undersea Hyperbaric Medical society*, vol. 30, no. 1, 2003, pp. 57-73.
- [3] P.D. Neuhaus, M. O'Sullivan, D. Eaton, J. Carff, J.E. Pratt, "Concept Designs for Underwater Swimming Exoskeletons," *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA '04)*, New Orleans, LA..
- [4] P.E. di Prampro, D.R. Pendergast, D.W. Wilson, and D.W. Rennie, "Energetics of swimming in man," *J. of Applied Physiology*, vol. 37, no. 1, July 1974.

III. Human-Agent Teamwork and Agile Computing

In the years ahead, unmanned systems will be used on an ever-increasing scale [19]. A key requirement for such systems is for real-time cooperation with people and with other autonomous systems. While these heterogeneous cooperating platforms may operate at different levels of sophistication and with dynamically varying degrees of autonomy, they will require some common means of representing and appropriately participating in joint tasks. Just as important, developers of such systems will need tools and methodologies to assure that such systems will work together reliably and safely, even when they are designed independently.

An equally challenging problem involves the fact that unmanned vehicles are subject to communication constraints that limit bandwidth and increase latency. In addition, network disconnection is a concern, whether due to vehicles moving out of communications range, communications being obstructed by terrain, or a tactical need to minimize signal transmissions. Finally, communication may sometimes depend on peer-to-peer networks, where one vehicle communicates with another vehicle by using a third vehicle as a relay. These problems are particularly acute for undersea and surf-zone environments.

Both the dynamics of human-autonomous system coordination and the ongoing management of real-time operational constraints can be addressed by the use of software agent technology. Software agents are loosely-coupled components designed with a variety of built-in communicative and collaborative capabilities. In addition to these built-in generic capabilities, each agent usually serves as a package for some more specific intelligent functionality (e.g., sensing, fusion, analytic, or navigation behavior). The combination of these generic and agent-specific capabilities help enable unmanned vehicles to function as effective “team members” with each other and with other autonomous systems. Under the strict control of administrator-defined policies, one or more software agents may be permitted to populate a given hardware vehicle platform or to move around the network as needed under their own power, operating in dynamically optimized onboard or off-board combinations.

The combination of human-agent teamwork and agile computing capabilities afford a degree of flexibility and responsiveness in the configuration and tasking of unmanned vehicles that goes far beyond what is possible with today’s technology. Different tasks and missions place different requirements on the unmanned vehicles and, given their limited processing and storage capabilities, the necessary algorithms for responding to dynamically changing conditions will often need to be pushed to the vehicles in real time. Changing conditions may require adaptive task allocation among humans and machines, including a requirement that other nearby resources may need to be rapidly discovered for immediate exploitation. If a human team member becomes disabled or a vehicle is suddenly destroyed, the survivability of the system depends directly on being able to quickly shift tasks and capabilities among people and platforms consistent with pre-approved operating policies and procedures.

In this research focus, we are addressing these issues through the development of policy-based human-agent teamwork and agile computing infrastructures. These developments will result in a robust teamwork-aware computational infrastructure for unmanned systems that is secure, reliable, and capable.

Human-Agent Teamwork

Background

KAoS Policy and Domain Services. The increased intelligence afforded by autonomous systems is both a boon and a danger. By their ability to operate independently without constant human supervision, they can perform tasks that would be impractical or impossible using conventional platforms. On the other hand, this additional autonomy, if unchecked, also has the potential of effecting severe damage in the case of buggy or malicious software. Because ever more powerful intelligent agents will increasingly differ in important ways from conventional software of the past, we need to take into account the social issues no less than the technical ones if the agents we design and build are to be acceptable to people. Writes Don Norman:

“The technical aspect is to devise a computational structure that guarantees that from the technical standpoint, all is under control. This is not an easy task. The social part of acceptability is to provide reassurance that all is working according to plan... This is [also] a non-trivial task” [17, p. 51].

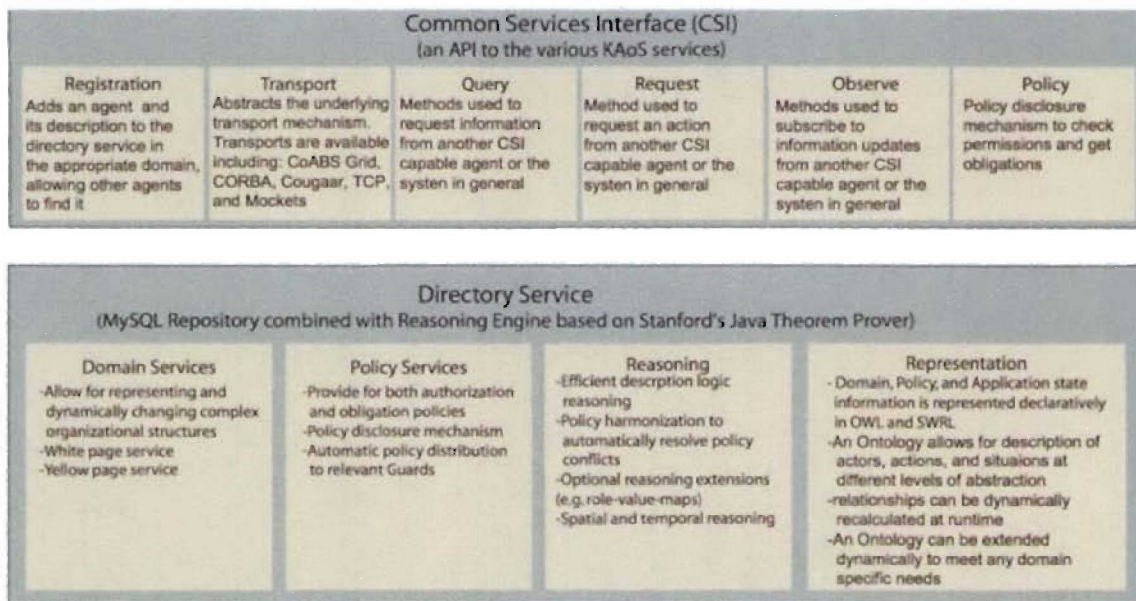
The objective of KAoS is to address some of the technical and social aspects of agent design for increased human acceptability through a policy-based approach. From a technical perspective, we want to be able to help ensure the protection of agent state, the viability of agent communities, and the reliability of the resources on which they depend. To accomplish this, we must guarantee insofar as possible that the autonomy of agents can always be bounded by explicit enforceable policy that can be continually adjusted to maximize their effectiveness and safety in both human and computational environments.

From a social perspective, we want agents to be designed so as to fit well with how people actually work together. Explicit policies governing human-agent interaction based on careful observation of work practice and an understanding of current social science research can help assure that effective and natural coordination, appropriate levels and modalities of feedback, and adequate predictability and responsiveness to human control are maintained. These factors are key to providing the reassurance and trust that are prerequisite to the widespread acceptance of agent technology for non-trivial applications.¹

KAoS a collection of componentized policy and domain management services compatible with several popular agent frameworks, including Nomads [20], the DARPA

¹ A more complete study of many of these topics can be found in [5; 9].

CoABS Grid [14], the DARPA ALP/Ultra*Log Cougaar framework (<http://www.cougaar.net>), CORBA (<http://www.omg.org>), Voyager (<http://www.recurSIONsw.com/osi.asp>), Brahms (www.agentisolutions.com), TRIPS [1; 4], and SFX (<http://crasar.eng.usf.edu/research/publications.htm>). While initially oriented to the dynamic and complex requirements of software agent applications, KAoS services are also being adapted to general-purpose grid computing (<http://www.gridforum.org>), the Joint Battlespace Infosphere (JBI; <http://www.rl.af.mil/programs/jbi/>), and Web Services (<http://www.w3.org/2002/ws/>) environments as well [13; 25]. KAoS has been deployed in a wide variety of applications, from coalition warfare [10; 21] and agile sensor feeds [22], to process monitoring and notification [11], to robustness and survivability for distributed systems [15], to semantic web services composition [25], to human-agent teamwork in space applications [9], to cognitive prostheses for augmented cognition [5; 18]. The adaptability of KAoS is due in large part to its pluggable infrastructure based on Sun's Java Agent Services (JAS) (<http://www.java-agent.org>). For a full description of KAoS, the reader is referred to [2; 6; 7; 10; 24; 26].

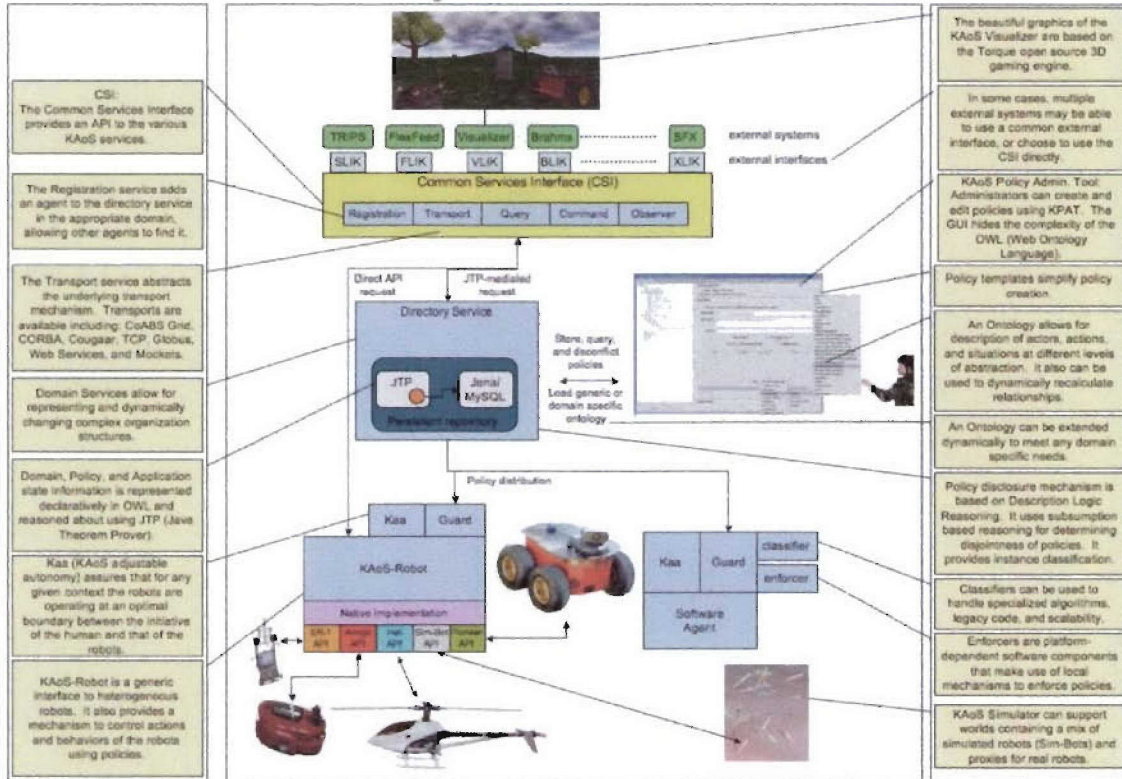


KAoS Common Services Interfaces and Directory Service

KAoS domain services provide the capability for groups of agents to be structured into organizations of agent domains and subdomains to facilitate human-agent collaboration and external policy administration. Domains may represent any sort of group imaginable, from potentially complex organizational structures to administrative units to dynamic task-oriented teams with continually changing membership. A given domain can extend across host boundaries and, conversely, multiple domains can exist concurrently on the same host. Domains may be nested indefinitely and, depending on whether policy allows, agents may become members of more than one domain at a time.

KAoS policy services allow for the specification, management, conflict resolution, and enforcement of policies within domains [10; 16; 26]. Policies are standard OWL (W3C's Web Ontology Language) specifications that constrain the performance of some type of

action by one or more actors in a given situation [23].² The policy ontology distinguishes between authorizations (i.e., constraints that permit or forbid some action) and obligations (i.e., constraints that require some action to be performed, or else serve to waive such a requirement). Through various property restrictions in the action type, a given policy can be variously scoped, for example, either to individual agents, to agents of a given class, to agents belonging to a particular group, or to agents running in a given physical place or computational environment (e.g., host, VM).



Components and features of KAoS as used in the NAIMT application

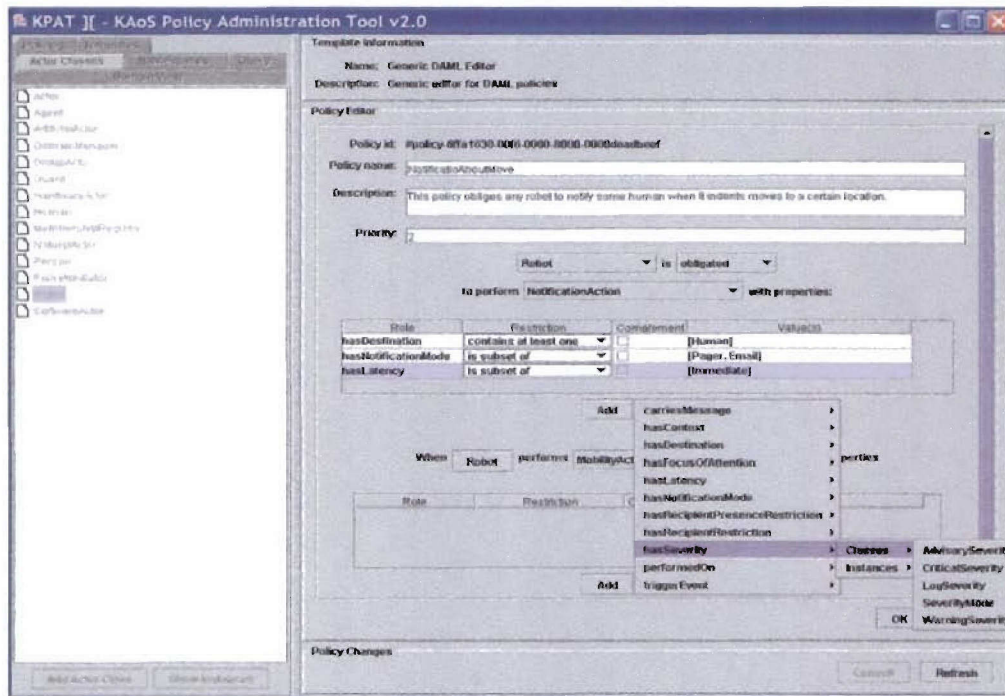
The KAoS Policy Administration Tool (KPAT³) implements a graphical user interface to policy and domain management functionality. It has been developed to make policy management easier for administrators without requiring extensive training. Using KPAT, an authorized user may make changes to policy or domain structure from anywhere using a secure Web browser. Alternatively, trusted infrastructure components such as Guards may, if authorized, propose policy changes autonomously or semi-autonomously based on their observation of system events.

KPAT can be used to browse and load ontologies, to define, deconflict, and commit new policies, and to modify or delete old ones. Groups of interdependent policies can be composed into policy sets. The generic OWL Policy Editor is a powerful view that allows

² Where expression of a policy require going beyond description logic, judicious extensions to the semantics are possible within KAoS (e.g., role-value maps [27]).

³ Pronounced KAY-pat.

administrators fine-grained control over any aspect of policy specification. It is driven by the ontologies loaded into JTP and its constraint-driven interface always provides the user with the list of choices narrowed to only those appropriate to the context of the other current selections. Custom editors tailored to particular kinds of policies may also be added to KPAT and will be automatically invoked by default if a policy about the action class associated with the given custom editor is selected for editing. When a user commits a change to an ontology (e.g., a new or edited policy, changes to domain structure) the Jena framework (<http://www.hpl.hp.com/semweb/>) is used to dynamically build a OWL representation based on the values selected by the user.

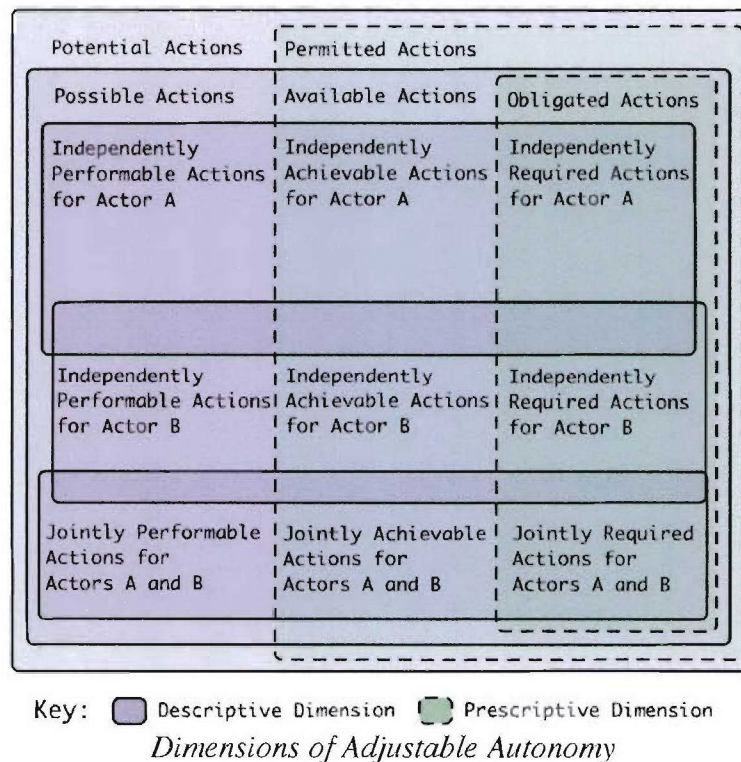


KPAT shown notifying the user of the results of policy harmonization.

In conjunction with the Mixed-Initiative Human Control team, we are currently conducting research to develop and evaluate formalisms and mechanisms for adjustable autonomy and policies that will facilitate mixed-initiative interaction [7]. On the foundation of current policy mechanisms built to support human users, we are building *Kaa* (KAoS adjustable autonomy) a component that permits KAoS to perform self-adjustments of autonomy consistent with policy [8].

The objective of *Kaa* is to be able to reason about and automatically or semi-automatically adjust autonomy along whichever dimensions (possibility, performability, authorization, obligation) are deemed to provide the most effective result. To the extent circumstances allow *Kaa* to adjust agent autonomy with reasonable dynamism (ideally allowing handoffs of control among team members to occur anytime) and with a sufficiently fine-grained range of levels, teamwork mechanisms can flexibly renegotiate roles and tasks among humans and agents as needed when new opportunities arise or

when breakdowns occur. Such adjustments can also be anticipatory when agents are capable of predicting the relevant events [3; 12].



Work Performed

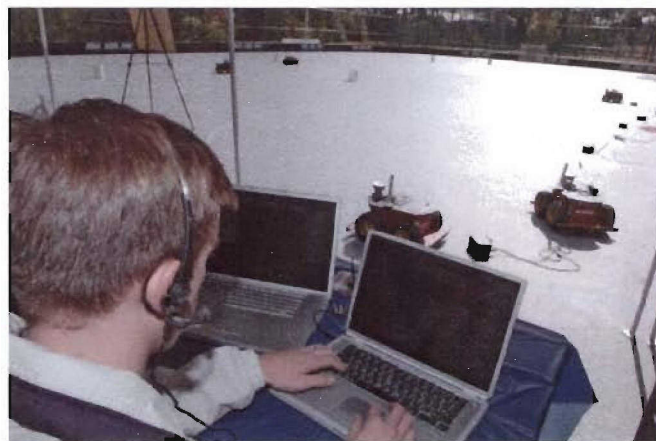
During year 1 we performed the following tasks:

- **Obligation policy support in KAoS.** We have implemented an initial version of support in KAoS for obligation policies (i.e., constraints that require or waive requirements for certain kinds of actions in a given context), with enablers to enforce them and enhancements to the KPAT user interface to define them. We have also developed an initial implementation of KAoS Robot interfaces, and have begun development of simulation capabilities and viewers. We have converted KAoS policy representations from DAML to OWL.
- **Safe and secure autonomous operation.** We incorporated an initial KAoS policy enforcement mechanism into the agile computing bandwidth management component and have demonstrated an initial version of these capabilities. We have purchased robotic hardware (in consultation with USF so that we have compatible setups) and have established an initial UAV/UGV robotic testbed at IHMC. We have begun integration of our components with USF's in collaboration on the distributed field robot architecture.
- **Effective and natural human-agent interaction.** We performed an initial study of what display and behavior options people find most effective for robots to communicate common states and actions. We developed an initial set of ontologies and notification and event policy implementation components. We

have begun to integrate KAoS with the dialogue system components and have demonstrated these capabilities in conjunction with physical robots.

In the second year, we leveraged our previous work to extend our combined systems capabilities in an application using real robots in an outdoor environment. The scenario chosen to validate our ideas was lane finding in a littoral environment; an important issue to our Navy collaborators. The premise is that an amphibious landing is planned in an area with suspected mines. The basic idea is to search the area to find a path or lane wide enough for safe transit of landing troops. Our demonstration simulated the scenario using a large parking lot with mines (buckets) and other obstacles (cones, boxes, garbage cans, etc.). We used up to four Pioneer 3-AT robots to perform the search. All platforms were controlled by a single user through the TRIPS multi-modal dialogue system and communication was via a dynamic ad hoc network provided by IHMC's mockets implementation. One Raptor 30 helicopter occasionally supplemented the search as a communications relay. We held demonstrations at both IHMC and NSWC-PC. Our demonstration highlighted several major areas:

- Matchmaking and Mission Assignment
- Mission Execution Teamwork
- Mixed-Initiative Classification
- Dynamic policy creation and application
- Adjustable autonomy
- Policy governed proactive network maintenance
- Opportunistic resource exploitation

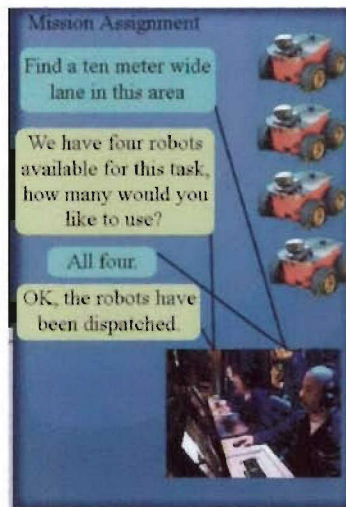


NSWC-PC demo

Matchmaking and Mission Assignment

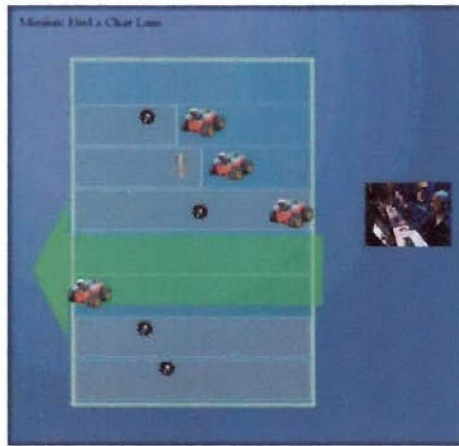
We have moved away from the traditional static one-to-one matching of operator to robot and provide dynamic team formation by reasoning about resource capabilities and providing matchmaking between tasks and resources. A single operator interacted with the TRIPS dialogue system, using a combination of natural language and graphical input. To start the mission, the operator highlighted an area on the screen and made a request. Each resource registered with KAoS and advertised its capabilities. When a requested task is generated through TRIPS, KAoS can find the available resources whose

capabilities match the requirements of the given task. TRIPS informs the user of available resources and the users make their selection. TRIPS then uses KAoS to dispatch commands to the selected platforms.



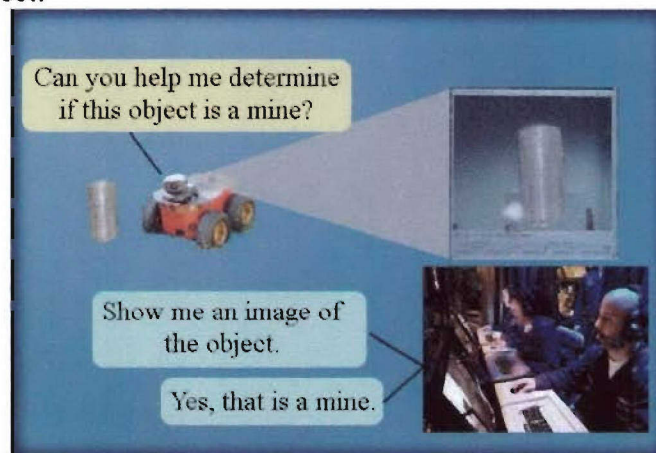
Mission Execution Teamwork

During mission execution we did not use predefined roles and tasks, but instead we coordinated joint activity within the distributed KAoS-Robot architecture based on current capability and availability to satisfy team goals and provide autonomous failure recovery. Once the robots had been dispatched, they collaborated to divide the area and begin the task. The robot chooses a non-searched leg of the area, moves to the leg and begins searching using an algorithm developed by NSWC-PC. The robots use sonar to find obstacles and a camera to classify them. If the object is determined to be a mine, the leg is aborted and the team members are updated with the information that the leg is obstructed by a mine. If the object is not a mine, the robot avoids the obstacle and continues searching the leg. If the leg is clear, the robot selects another leg to search and updates the team. If a clear lane of the specified width has been found or there are no more legs to search, the mission is complete and the robots return to base. Failure of a single robot does not prevent completion of the mission as the remaining robots will coordinate to complete the mission. Robots may also be temporarily taken off task in support of communications relay as described later. Upon completion, they can resume their search task, coordinating with the team to ensure appropriate collaboration.



Mixed Initiative Classification

Mixed-initiative interaction is a shift away from the model where a human directs the robot and the robot complies. It allows the human and robot to interact flexibly and naturally. Each can direct the interaction based on their abilities and needs. We showed how a robot can work with a human to classify indeterminate objects through an interface that combines graphics and dialogue as appropriate. Dialogue is handled through the TRIPS system. Normally the robots take the initiative to classify objects on their own, but when they are uncertain about the identity of an object they can request assistance from a human. Using the TRIPS interface in conjunction with KAoS-Robot capabilities operator can request a video image, adjust the camera angle and zoom, and then assist the robot in classifying the object.



Dynamic policy creation and application

Most previous work attempts to code all limitations and constraints into each robot a-priori and is not flexible at run-time. Our work allows for dynamic creation and application of constraints on robotic platforms at an individual or group level. As policies are created they can be checked for consistency to identify conflicts early. To demonstrate the dynamic policy creation and application we use a simple restricted area policy. The robots freely move through the restricted area at the beginning of the search when the policy does not exist. We use the KAoS Policy Administration Tool (KPAT,

pronounced KAY-pat) to create the policy. Once created, the policy is automatically distributed and becomes applicable. The policy delineates a staging area for the troops that the robots are not allowed to enter. As the robots try to return to base, the new policy is now effective and requires the robots to alter their path to avoid the restricted area as shown in the figure below. This is just one simple example of how policy can be used to manage robot behavior without requiring changes to the robot code or even explicit commands to the robot. The policy is completely external to the particular robots implementation and is viewable by any users working with the robots.

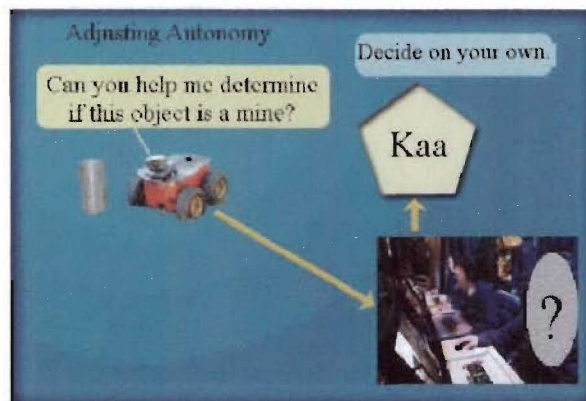


Adjustable Autonomy

We view Adjustable Autonomy as more than just changing the control mode along the spectrum from teleoperation to autonomous operation. It assures that robots are continuously operating at the optimal boundary between the initiative of the human and that of the robot. We use decision-theoretic algorithms to determine if some dimension of robot autonomy should be adjusted in order to prevent task failure and have the ability to impose and modify constraints that affect the range of actions available to an agent. As part of this project, we have developed the KAoS adjustable autonomy component (Kaa). Kaa monitors the current situation and reasons about what kind of adjustment choices are available to improve performance. Normally, when a robot fails to classify an object, it is obliged to get assistance from a human, an action which is typically performable. However, if the operator is attending to another robot's request, the robot will wait a specific period and then timeout causing failure of the attempt to get assistance. The non-availability of the human made this task non-performable, even though it is obligated. For our current scenario, Kaa considers the utility of adjustments to policy constraints based on safety, timeliness, and completeness. Kaa has several adjustment choices. One is to remove the obligation, but in this case the performance is unaffected since the classification will still fail. Another option is to increase performability. Kaa has knowledge unavailable to individual agents, such as the status of the human, and can use this knowledge to assist in such situations. Kaa obliges the robot to continue waiting until the operator is no longer engaged with the other robot, thus making the action performable again and improving performance, under the assumption that waiting for the human will result in a more accurate classification. Kaa's role becomes increasingly important as the complexity of the team, task and environment increase.

Policy Governed Proactive Network Maintenance

We have advanced beyond the standard single hop network techniques to be able to proactively move resources in support a multi-hop ad hoc network. We also can provide filtering and transformation of data along the path, all governed by KAoS policies. Occasionally robots may move out of range and lose communications. As the robot moves out of range, the connectivity will fail. And the robot will stop. If the robot were to simply move back into range communications would be restored, but our goal was to extend the range beyond single hop communications by recognizing the failure and attempting to proactively restore communications by trying to move an available resource. KAoS policies are used to manage the use of resources for communications relay by Flexfeed. There is a policy in place that prohibits the use of resources as a communication relay, which initially prevents the robot from accepting the task. However, Kaa determines the critical nature of restoring communications and attempts to provide a one-time override of the policy. This change increases the allowable actions of the robot and enables the use of the robot as a communications relay. Once Kaa has overridden the policy, the robot is authorized to be a communications relay, so the Flexfeed request can be executed on the robot. The robot moves into the most likely position to provide connectivity. As the connectivity is restored, the robot that had lost communications is able to continue searching.



Opportunistic resource exploitation

Instead of having a rigid network, we were able to opportunistically leverage resources. Available resources can be dynamically detected and manipulated to leverage all available computational power and network bandwidth. Once communications have been restored, our search capability was reduced by one robot. We introduce a new platform not engaged in the search, but capable of filling a role of communications relay. As the helicopter enters the area, a communications path is automatically established by the Flexfeed infrastructure. The original communications relay robot can detect that it is no longer needed and is free to continue searching. In this way, we demonstrated flexibility to dynamically detect changes in the communications environment and automatically adapt to maximize the effective for a given task.



Helicopter providing communications relay for ground robots

References

- [1] Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2001). Towards conversational human-computer interaction. *AI Magazine*, 22(4), 27-35.
- [2] Allsopp, D., Beautement, P., Kirton, M., Tate, A., Bradshaw, J. M., Suri, N., & Burstein, M. H. (2003). The Coalition Agents Experiment: Network-Enabled Coalition Operations. *Journal of Defence Science*, 8(3), 130-141.
- [3] Boella, G. (2002). Obligations and cooperation: Two sides of social rationality. In H. Hexmoor, C. Castelfranchi, & R. Falcone (Ed.), *Agent Autonomy*. (pp. 57-78). Dordrecht, The Netherlands: Kluwer.
- [4] Bradshaw, J. M., Acquisti, A., Allen, J., Breedy, M. R., Bunch, L., Chambers, N., Feltovich, P., Galescu, L., Goodrich, M. A., Jeffers, R., Johnson, M., Jung, H., Lott, J., Olsen Jr., D. R., Sierhuis, M., Suri, N., Taysom, W., Tonti, G., & Uszok, A. (2004). Teamwork-centered autonomy for extended human-agent interaction in space applications. *AAAI 2004 Spring Symposium*. Stanford University, CA, AAAI Press.
- [5] Bradshaw, J. M., Beautement, P., Breedy, M. R., Bunch, L., Drakunov, S. V., Feltovich, P. J., Hoffman, R. R., Jeffers, R., Johnson, M., Kulkarni, S., Lott, J., Raj, A., Suri, N., & Uszok, A. (2004). Making agents acceptable to people. In N. Zhong & J. Liu (Ed.), *Intelligent Technologies for Information Analysis: Advances in Agents, Data Mining, and Statistical Learning*. (pp. 361-400). Berlin: Springer Verlag.
- [6] Bradshaw, J. M., Dutfield, S., Benoit, P., & Woolley, J. D. (1997). KAoS: Toward an industrial-strength generic agent architecture. In J. M. Bradshaw (Ed.), *Software Agents*. (pp. 375-418). Cambridge, MA: AAAI Press/The MIT Press.
- [7] Bradshaw, J. M., Feltovich, P., Jung, H., Kulkarni, S., Taysom, W., & Uszok, A. (2004). Dimensions of adjustable autonomy and mixed-initiative interaction. In M. Nickles, M. Rovatsos, & G. Weiss (Ed.), *Agents and Computational Autonomy*:

- Potential, Risks, and Solutions. Lecture Notes in Computer Science, Vol. 2969.* (pp. 17-39). Berlin, Germany: Springer-Verlag.
- [8] Bradshaw, J. M., Jung, H., Kulkarni, S., Johnson, M., Feltovich, P., Allen, J., Bunch, L., Chambers, N., Galescu, L., Jeffers, R., Suri, N., Taysom, W., & Uszok, A. (2005). Toward trustworthy adjustable autonomy in KAoS. In R. Falcone (Ed.), *Trusting Agents for Trustworthy Electronic Societies*. (pp. in press). Berlin: Springer.
 - [9] Bradshaw, J. M., Sierhuis, M., Acquisti, A., Feltovich, P., Hoffman, R., Jeffers, R., Prescott, D., Suri, N., Uszok, A., & Van Hoof, R. (2003). Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications. In H. Hexmoor, R. Falcone, & C. Castelfranchi (Ed.), *Agent Autonomy*. (pp. 243-280). Kluwer.
 - [10] Bradshaw, J. M., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M. H., Acquisti, A., Benyo, B., Breedy, M. R., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., & Van Hoof, R. (2003). Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003)*. Melbourne, Australia, New York, NY: ACM Press.
 - [11] Bunch, L., Breedy, M. R., & Bradshaw, J. M. (2004). Software agents for process monitoring and notification. *Proceedings of AIMS 04*.
 - [12] Falcone, R., & Castelfranchi, C. (2002). From automaticity to autonomy: The frontier of artificial agents. In H. Hexmoor, C. Castelfranchi, & R. Falcone (Ed.), *Agent Autonomy*. (pp. 79-103). Dordrecht, The Netherlands: Kluwer.
 - [13] Johnson, M., Chang, P., Jeffers, R., Bradshaw, J. M., Soo, V.-W., Breedy, M. R., Bunch, L., Kulkarni, S., Lott, J., Suri, N., & Uszok, A. (2003). KAoS semantic policy and domain services: An application of DAML to Web services-based grid architectures. *Proceedings of the AAMAS 03 Workshop on Web Services and Agent-Based Engineering*. Melbourne, Australia.
 - [14] Kahn, M., & Cicalese, C. (2001). CoABS Grid Scalability Experiments. O. F. Rana (Ed.), *Second International Workshop on Infrastructure for Scalable Multi-Agent Systems at the Fifth International Conference on Autonomous Agents*. Montreal, CA, New York: ACM Press.
 - [15] Lott, J., Bradshaw, J. M., Uszok, A., & Jeffers, R. (2004). Using KAoS policy and domain services within Cougaar. *Proceedings of the Open Cougaar Conference 2004*, (pp. 89-95). New York City, NY.
 - [16] Lupu, E. C., & Sloman, M. S. (1999). Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering—Special Issue on Inconsistency Management*.
 - [17] Norman, D. A. (1997). How might people interact with agents? In J. M. Bradshaw (Ed.), *Software Agents*. (pp. 49-55 (see also How might people interact with robots? http://www.jnd.org/dn.mss/how_might_humans_int.html)). Cambridge, MA: The AAAI Press/The MIT Press.
 - [18] Raj, A., Bradshaw, J. M., Carff, R. W., Johnson, M., & Kulkarni, S. (2004). An agent-based approach for AugCog integration and interaction. *Proceedings of Augmented Cognition: Improving Warfighter Information Intake Under Stress*. Orlando, FL,

- [19] Summey, D. C., Rodrigues, R. R., DeMartino, D. P., Portmann Jr., H. H., & Moritz, E. (2001). *Shaping the Future of Naval Warfare with Unmanned Systems*. CSS/TS-01/09. Dahlgren Division, Naval Surface Warfare Center, July.
- [20] Suri, N., Bradshaw, J. M., Breedy, M. R., Groth, P. T., Hill, G. A., Jeffers, R., Mitrovich, T. R., Pouliot, B. R., & Smith, D. S. (2000). NOMADS: Toward an environment for strong and safe agent mobility. *Proceedings of Autonomous Agents 2000*. Barcelona, Spain, New York: ACM Press,
- [21] Suri, N., Bradshaw, J. M., Burstein, M. H., Uszok, A., Benyo, B., Breedy, M. R., Carvalho, M., Diller, D., Groth, P. T., Jeffers, R., Johnson, M., Kulkarni, S., & Lott, J. (2003). DAML-based policy enforcement for semantic data transformation and filtering in multi-agent systems. *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003)*. Melbourne, Australia, New York, NY: ACM Press,
- [22] Suri, N., Bradshaw, J. M., Carvalho, M., Breedy, M. R., Cowin, T. B., Saavendra, R., & Kulkarni, S. (2003). Applying agile computing to support efficient and policy-controlled sensor information feeds in the Army Future Combat Systems environment. *Proceedings of the Annual U.S. Army Collaborative Technology Alliance (CTA) Symposium*.
- [23] Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., & Uszok, A. (2003). Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. In D. Fensel, K. Sycara, & J. Mylopoulos (Ed.), *The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, USA, October 2003, LNCS 2870*. (pp. 419-437). Berlin: Springer.
- [24] Uszok, A., Bradshaw, J. M., & Jeffers, R. (2004). KAoS: A policy and domain services framework for grid computing and semantic web services. *Proceedings of the Second International Conference on Trust Management*. Oxford, England,
- [25] Uszok, A., Bradshaw, J. M., Jeffers, R., Johnson, M., Tate, A., Dalton, J., & Aitken, S. (2004). Policy and contract management for semantic web services. *AAAI 2004 Spring Symposium Workshop on Knowledge Representation and Ontology for Autonomous Systems*. Stanford University, CA, AAAI Press,
- [26] Uszok, A., Bradshaw, J. M., Jeffers, R., Suri, N., Hayes, P., Breedy, M. R., Bunch, L., Johnson, M., Kulkarni, S., & Lott, J. (2003). KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. *Proceedings of Policy 2003*. Como, Italy,
- [27] Uszok, A., Bradshaw, J. M., Johnson, M., Jeffers, R., Tate, A., Dalton, J., & Aitken, S. (2004). KAoS policy management for semantic web services. *IEEE Intelligent Systems*, 19(4), 32-41.

Agile Computing

Background

Agile computing is an innovative metaphor for distributed computing systems and prescribes a new approach to their design and implementation.

Agile computing may be defined as opportunistically discovering, manipulating, and exploiting available computing and communication resources in order to improve capability, performance, efficiency, fault-tolerance, and survivability. The term **agile** is used to highlight the desire to both quickly react to changes in the environment as well as to take advantage of transient resources only available for short periods of time. Agile computing thrives in the presence of highly dynamic environments and resources, where nodes are constantly being added, removed, and moved, resulting in intermittent availability of resources and changes in network reachability, bandwidth, and latency.

The notion of agile computing builds on current research in grid computing, mobile agents, ad-hoc networking, and peer to peer resource sharing. The specific realization described here exploits mobility of code, data, and computation and an architecture independent uniform execution environment to achieve the desired property of agility.

Technical Requirements and Challenges

Resource Discovery

Resource discovery is one of the cornerstones of agile computing and is fundamental to opportunistically finding resources. The resource discovery mechanism is currently based on a UDP broadcast mechanism and works over most wireless network environments based on 802.11. Other approaches can be developed as needed to support different networking environments. The overall goal is to take advantage of any available and appropriate discovery services such as Bluetooth, Salutation Protocol and SLP.

Group Formation

A group is defined as a set of systems that specify the scope for resource sharing. Therefore, formation and regulation of groups is central to agile computing. Groups may be formed through static configuration or through ad-hoc discovery. For example, all of the robots performing a search task may be configured to be part of a group. On the other extreme, a group may be formed when four laptop computers in a meeting room discover each other through a protocol such as Bluetooth. Group formation is controllable through policies for security reasons so that the systems that are allowed to share resources can be regulated.

Architecture Independence

Agile computing implies that computations must be able to take advantage of any available resources independent of the underlying hardware architecture. If a system fails unexpectedly, the infrastructure should be able to compensate by running those computations on another system, independent of architecture. Similarly, if an ad-hoc group consists of nodes of different architectures (say, robots with embedded computers, PDAs, and laptops running Windows and Linux), any one of the nodes should still be able to take advantage of any available resources in the other nodes.

Environmental Independence

Architecture independence alone is not sufficient to support migration of computations between systems. If the environment on each system is different, the computation will

fail after migration due to the sudden change. Environmental factors include the data resident on a system, configuration of the software on the system as well as any specialized hardware in the system.

Mobility of Code, State, and Data

One of the key enhancements provided by agile computing over existing approaches is the exploitation of mobility of code, computational state, and data. Mobility of code is needed to support dynamically pushing or pulling code to a system in order to change its configuration or download a new capability. Mobility of computational state is needed to allow computations to move from one system to another within a group. Such movement may be triggered for survivability, performance, or to accommodate changes in group membership. Finally, mobility of data with the computation and/or the code is necessary to handle potential network disconnections as well as to optimize network bandwidth usage.

Security

Security is of paramount importance for agile computing to be used in a practical environment. A system that joins a group and contributes resources should be protected from abuse. If computations are pushed onto a system, the computations must be executed in a restricted environment to guarantee that they do not access private areas of the host system or abuse the resources available on the host. Similarly, the computations themselves need to be protected from a possibly malicious host that joins a group.

In addition to satisfying the above requirements the following two challenges must be addressed to make agile computing successful:

Overcoming Dedicated Roles / Owners for Systems

One of the problems with current systems is that they are often dedicated to certain tasks or assigned to particular users. Such a priori classification of systems prevents exploitation of available resources. Agile computing relies on the notion that any available resource should be utilizable. In order to make this a reality, hardware should be generic and ubiquitous with the specialization being derived through software. If this were the case, then one system can easily be substituted for another by means of moving the software functionality as needed.

Similarly, if systems are assigned to individual owners who are protective about their systems, then resource sharing will be ineffective. One solution to this problem lies in resource accounting, which will allow the owner to a system to contribute resources but then to quantify the contribution in order to receive compensation in some manner.

Achieving a High Degree of Agility

The degree of agility may be defined as the length of time for which a system needs to be part of a group in order for its resources to be effectively exploited. The shorter the length of time, the higher the degree of agility. The degree of agility that can be realized is a direct function of the overhead involved. When a system joins a group, there is overhead in the group reformation process, in setting up communication channels, and in moving

computations, code, and data to the system. Before a system leaves a group, there is potentially more overhead in moving active computations off of the system.

The degree of agility may also be defined in terms of the minimum time required in order to reconfigure when one or more systems are under threat. A system that has a higher degree of agility will be more survivable.

Figure 3.1 shows the stages that the infrastructure goes through when a resource becomes available. The duration of time for which the resource is actually available is $t_4 - t_0$. Initially, the infrastructure goes through a discovery phase (from t_0 to t_1), when the availability of the resource becomes known. Then, the infrastructure needs to setup the new resource for use, which takes time from t_1 to t_2 . Setup includes such activities as pushing new code to the new system and/or migrating computations to the new system.

Once the setup process is complete, the new resource is fruitfully utilized by the infrastructure. This time is represented from t_2 to t_3 . This is the actual length of time during which the new resource is being productively utilized.

Before the resource becomes unavailable, the infrastructure goes through a preparation phase, which typically involves moving computations out of the resource about to go offline. This phase takes time from t_3 to t_4 . Once the resource disappears, the infrastructure goes through a recovery phase (from t_4 to t_5). This recovery phase might involve activities such as finding other systems to distribute the computations that were removed from the resource that went offline.

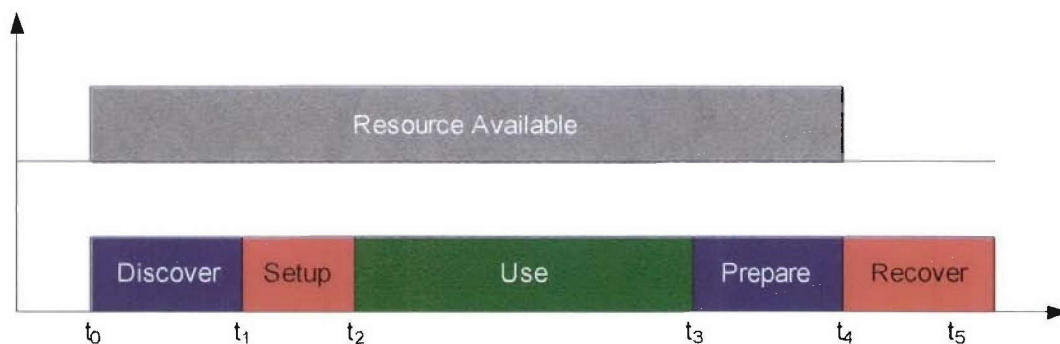


Figure 3.1: The Different Phases of Resource Utilization

In an ideal environment, the discovery, setup, preparation, and recovery times would be 0. The goal of the agile computing approach is to try and minimize these times as much as possible. The agile computing middleware being developed will help to better understand the current state of the art with respect to the overhead associated with each of these phases. An important analysis is to predict the impact of technology changes on each of these phases, thereby predicting the degree of agility of future systems.

Another interesting tradeoff involves preparation time versus recovery time. The expectation is that they are inversely proportional. That is, the greater the preparation

time, the lesser the recovery time. For a system to be highly survivable, the required time to prepare should be as close to 0 as possible.

System Components

The overall architecture for the agile computing framework consists of a kernel component, a coordination component, the mockets communications library, and the FlexFeed publish/subscribe library.

Agile Computing Kernel

Each participating node runs a specialized Java-compatible kernel (the Agile Computing Kernel) that provides a platform-independent execution environment and a set of local services such as policy enforcement and resource control.

The Agile Computing Kernel contains a uniform execution environment, a resource manager, an interface to a policy manager, a group manager, and a local coordinator. Figure 3.2 shows the main components of the Agile Computing Kernel. These components provide the set of capabilities that the running processes rely upon to take advantage of the agile computing framework. They constitute a middleware through which processes communicate and migrate between nodes. The following subsections provide a brief explanation of each of the components.

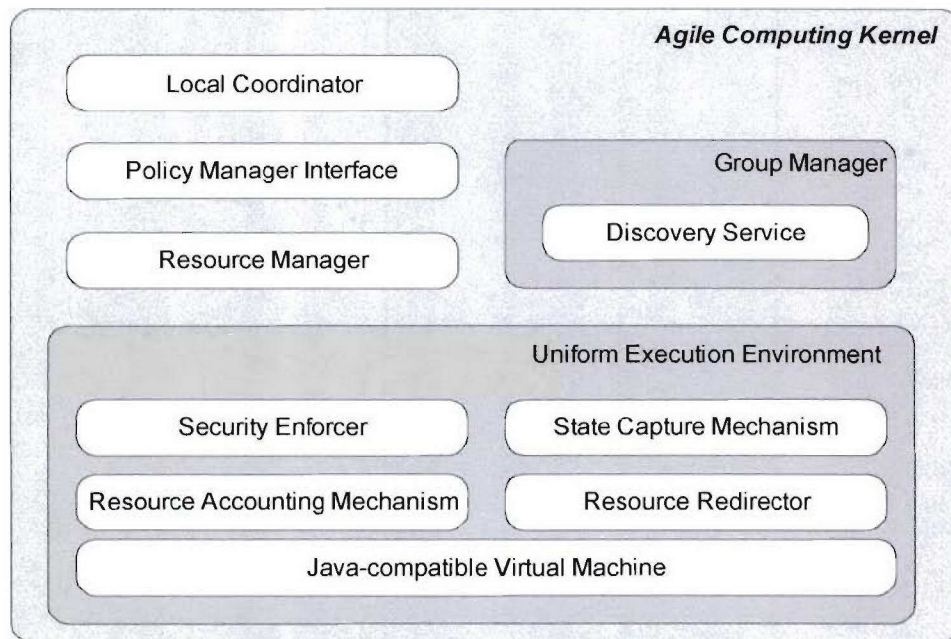


Figure 3.2: The Agile Computing Kernel

Uniform Execution Environment

The Uniform Execution Environment provides a common abstraction layer for code execution that hides underlying architectural differences such as CPU type and operating system. The execution environment supports the dynamic deployment and activation of

services, dynamic migration of services between kernels, secure execution of incoming services, resource redirection, resource accounting, and policy enforcement.

The implementation of the Uniform Execution Environment is currently based on either Aroma or the standard Java VM. Aroma is a clean-room implementation of a Java-compatible VM designed to provide architecture independence and support for agile computing requirements. Aroma was designed from the ground up to support capture of execution state of Java threads, provide accounting services for resource usage, and control resource consumption by Java threads. Moreover, the captured execution state is platform independent, which allows migration of computations between Aroma VMs that are running on different hardware platforms. The capabilities of the Aroma VM are critical to ensure secure execution of mobile code.

A standard Java VM is an alternative to the Aroma VM and provides higher performance. Adding resource control capabilities to the standard Java VM will provide a subset of the capabilities of Aroma. The JRaf2 framework is an example of a resource management framework for Java. A deployment of the middleware can contain any combination of kernels with Aroma and Java VMs, thereby providing additional flexibility.

There are no implicit or explicit requirements to use Java as the language for the implementation of the agile computing infrastructure. However, Java provides many desirable features for a mobile-code based framework. Moreover, the virtual machine architecture of Java provides platform independence.

Besides the Java-compatible VM, the execution environment also includes a set of software components that support interaction between the kernel and locally running services. These components are: a) the Security Enforcer, b) the Resource Accounting Mechanism, c) the State Capture Mechanism, and d) the Resource Redirector.

The Security Enforcer ensures that running services will have limited access to system resources to avoid denial of service (DOS) attacks. This component receives settings from the Policy Manager component specifying usage restrictions for each service running in the VM. The restrictions can be established during service deployment, migration, or even at runtime, after the service execution has started. This component also provides authentication and encryption mechanisms for secure data and state transfer.

The Resource Accounting Mechanism provides a facility to track resource utilization at the service level inside the VM. The mechanism is used by the Security Enforcer and the Resource Manager components to estimate overall kernel load and resource availability. The resource utilization profile of the service is also made available to the coordination component, which can use the information to decide optimal placement of services within the environment.

The Resource Redirector provides the means to transparently move links to local or remote resources when services are migrated between kernels. Consider, for example, a

scenario where a service has two network connections open to remote nodes. Due to an imminent power failure the service needs to move to another intermediate node. In this case, the Resource Redirector on each kernel will negotiate a redirection of the resources (the network connections in this case) to transparently move the service with no apparent interruption of the links. For the computation in this example, the migration happens seamlessly and the network connections with the remote hosts are maintained during the migration. The Resource Redirector implementation relies on the Mockets framework to provide migration of communication endpoints.

The State Capture Mechanism provides the necessary means to capture execution state of one or multiple services running in the execution environment. The execution state can be captured at any point between the execution of two Java bytecodes. The state information can then be persisted or moved to another host to resume execution on the very next bytecode. This capability is only available in conjunction with the Aroma VM and not the standard Java VM.

All these components work in concert with policies and the Resource Manager that are also part of the kernel but not directly integrated with the execution environment. The Resource Manager is primarily concerned with higher level interactions with the Coordinator and other kernels, but it does rely on the execution environment components to locally perform and enforce resource utilization policies.

Policy Manager Interface

The policy manager interface is responsible for communicating with the KAoS policy and domain services framework. KAoS manages the specification, conflict resolution, and distribution of policies to the enforcement components (in this case, the agile computing infrastructure). The interface component provides a facility for other components in the kernel to query and determine policies and restrictions that apply to local and remote services and nodes.

Resource Manager

The Resource manager provides an interface for the Coordinator and remote kernels to query and provide information about local resource utilization.

Resource availability is one of the metrics considered by the Coordinator when calculating optimum distribution of services. The Resource Manager acts as a bridge between the Accounting Service in the execution environment and the Coordinator. It monitors local resource utilization in the execution environment and interacts with the Coordinator to request migration of local services or to notify it of local resource availability for the Framework.

Group Manager

The Group Manager is the component responsible for identifying all the nodes that participate in a group. The framework is designed to handle highly dynamic environments, where nodes join and leave the framework at any arbitrary rate. Therefore,

a fundamental requirement is to efficiently and accurately identify other available nodes and services.

The role of the Group Manager is to coordinate with the Policy Manager to ensure proper advertisement of services and to identify and locate services required by local processes.

Although very useful for some types of applications, lookup services fail to provide important features that are necessary for agile computing. Some of these features are offered instead by the notion of service discovery.

The Group Manager relies on service discovery mechanisms, which provide a more general notion of service identification and location. In general, discovery protocols rely on the establishment of a global representation or state of the framework, available to each node. Unlike registry lookup, service discovery is an active service that announces the arrival and departure of service providers and capabilities. It doesn't necessarily rely on a centralized registry and it provides means to ensure service availability.

A dynamically formed group is a fundamental structural notion in agile computing. A group is essentially defined as a set of hosts that have joined together to share resources (for example, a set of laptops in a conference room during a meeting). Figure 3.3 shows one possible arrangement for a set of hosts. Note that groups may be disjoint or overlapping. An overlapping group is created by the existence of one or more shared hosts. A host may join multiple groups. Various grouping principles are possible but likely candidates are physical proximity, network reachability, and ownership.

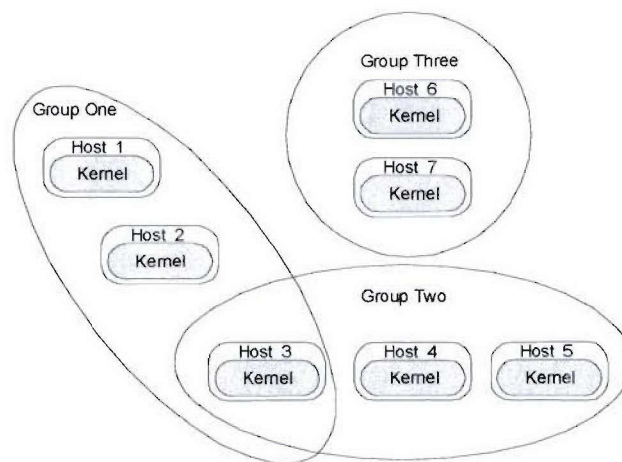


Figure 3.3: Runtime Grouping of Hosts

Hosts in a group might belong to different administrative domains, which creates another type of grouping. Domains are used to express common policies for hosts and to conveniently administer them. Domains tend to be more static compared to runtime groups. Figure 3.4 shows a configuration with two domains and one group.

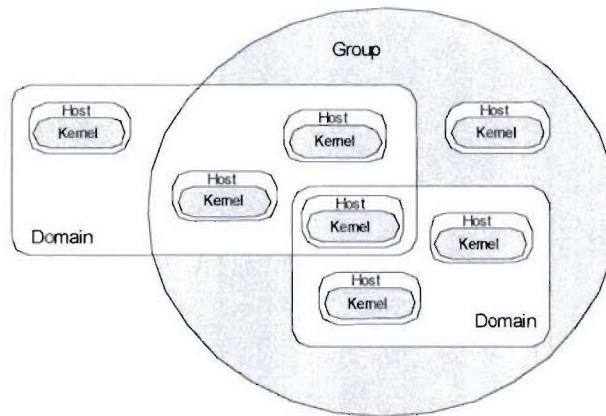


Figure 3.4: Relationship between Domains and Groups

Local Coordinator

The local coordinator works in conjunction with other local coordinators as well as centralized or zone-based coordinators to perform resource allocation, service deployment, service invocation, and service migration decisions. The coordination mechanism is discussed further in section the next section.

Coordinator

The coordinator is the logical entity that manages the overall behavior of the agile computing middleware. The coordinator monitors node and resource availability as well as network connectivity and bandwidth. Client requests are received by the coordinator, which handles allocation of resources. The coordinator also performs proactive manipulation of nodes, such as moving a node to act as a relay in order to restore a lost communications link. Coordination is a continuous process as the resource allocation needs to adapt to changes in the environment.

The coordinator may be realized using either a centralized approach, a zone-based approach, or a fully distributed approach. Figures 3.5 (a), 3.5 (b), and 3.5 (c) show the three different approaches.

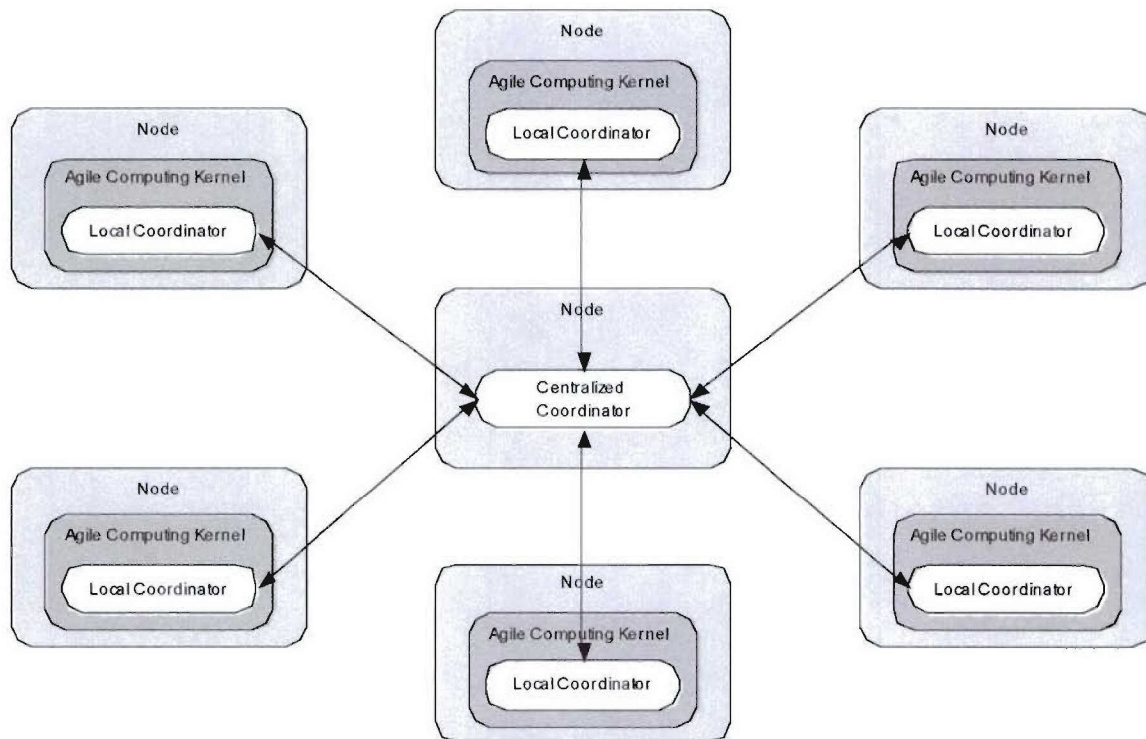


Figure 3.5 (a): Centralized Coordination Approach

In the centralized approach, a single node behaves as the coordinator at any given point in time. This coordinator communicates with the local coordinator at each of the nodes. Requests from clients are sent to the coordinator, which in turn issues commands to the other nodes. If the coordinator (or the node) fails, then a new coordinator can be selected through an election process.

Like any other centralized approach, the centralized coordinator is the simplest, but is not scalable and introduces a bottleneck as well as a single point of failure.

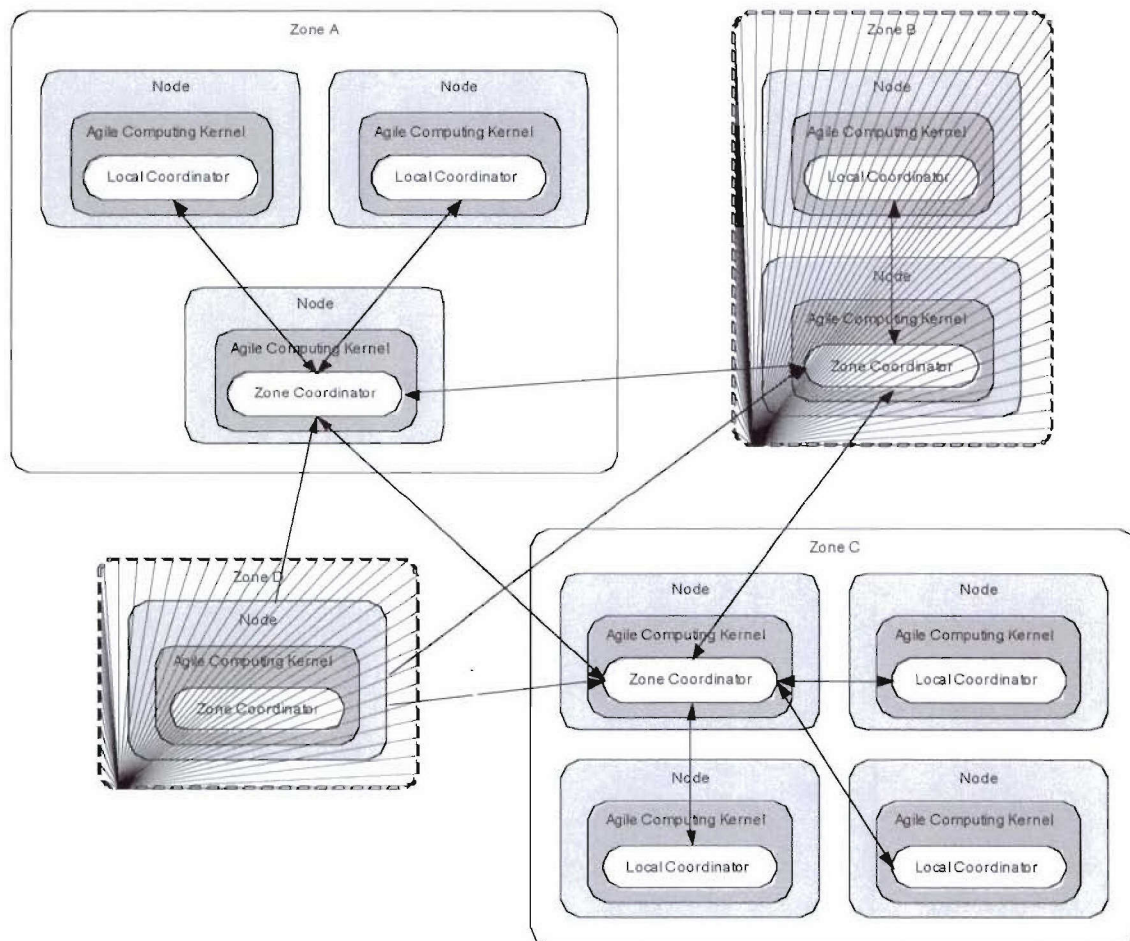


Figure 3.5 (b): Zone-based Coordination Approach

In the zone-based coordination approach, nodes are grouped into zones, each of which has the equivalent of a centralized coordinator. Zones are typically created based on network proximity (which, in wireless environments, also implies physical proximity). One of the nodes in a zone is elected to be the zone coordinator. This zone coordinator interacts with all the other local coordinators within the zone (much like the centralized approach) and also with other zone coordinators.

Two structural arrangements are possible with the zone-based approach – peer-to-peer and hierarchical. Figure 3.5(b) shows a fully connected peer-to-peer arrangement between the zones, although it is not necessary for all of the zone coordinators to be in direct communication with each other.

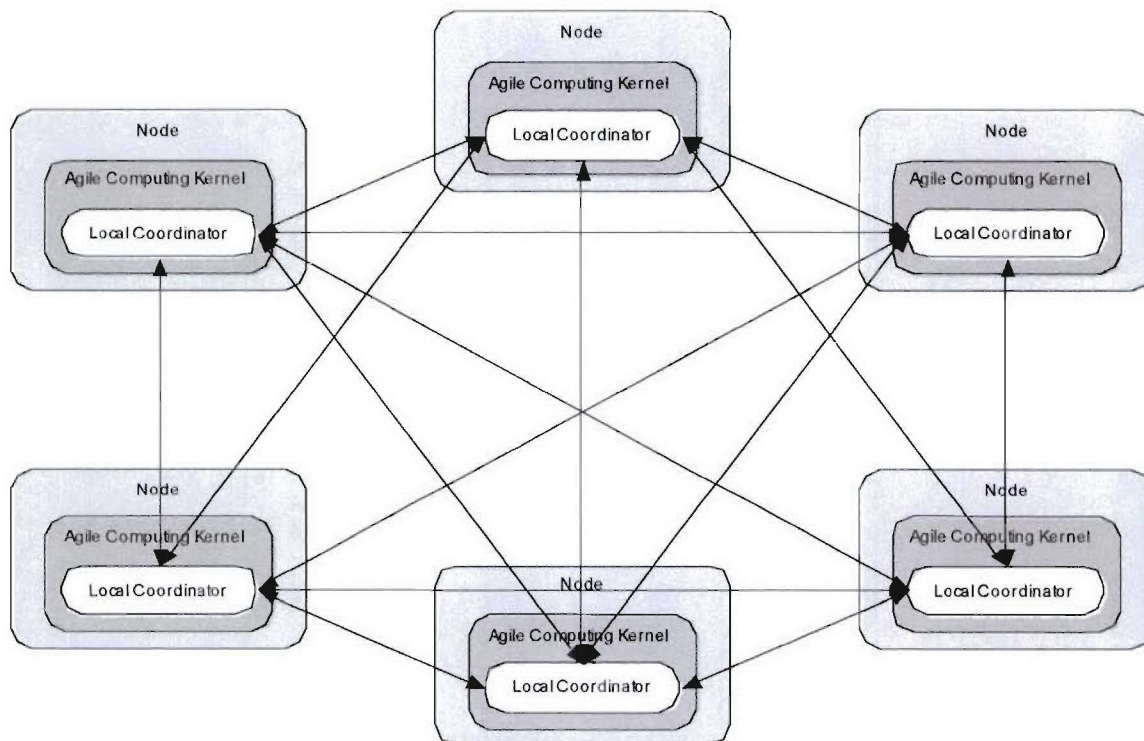


Figure 3.5 (c): Distributed Coordination Approach

The last possibility is a fully distributed coordination approach, as shown in Figure 3.5(c). In this approach, there is no centralized or partially centralized coordinator at all. Each of the local coordinators directly communicates with other local coordinators as needed. Again, Figure 3.5(c) shows a fully-connected arrangement, but that is not a requirement. The fully distributed coordination approach does not have a single point of failure but like most distributed algorithms, it is the most complicated approach.

In addition to the three different approaches, a number of different coordination algorithms are possible, based on the context and the problem to which the agile computing middleware is applied.

The overall goals and desired behavior of the coordinator can also be regulated via policies. Policies do not specify the coordination strategy, but rather runtime constraints on the strategy. For example, policies can be used to specify that only 50% of a node's CPU should be used, or that a node with less than 1 hour of battery life should not be exploited. Policies are specified using the KAoS framework, which allow the administrators or maintainers of the system to dynamically change behavior at runtime.

Mockets

Mockets (for "mobile sockets") is a comprehensive communications library for applications. The design and implementation of Mockets was motivated by the needs of tactical military information networks, which are typically wireless and ad-hoc with low bandwidth, intermittent connectivity, and variable latency. Mockets addresses specific challenges including the need to operate on a mobile ad-hoc network (where TCP does not perform optimally), provides a mechanism to detect connection loss, allows

applications to monitor network performance, provides flexible buffering, and supports policy-based control over application bandwidth utilization.

Mockets provides the following five features:

1. Application-level implementation of the communications library in order to provide flexibility, ease of distribution, and better integration between the application and the communications layer.
2. A TCP-style reliable, stream-oriented service that is designed to operate on wireless ad-hoc networks thereby making it easy to port existing applications to the ad-hoc environment.
3. A message-oriented service that provides enhanced capabilities such as message tagging and replacement, different classes of service (reliable/unreliable combined with sequenced/unsequenced), and prioritization.
4. Transparent mobility of communication endpoints from one host to another in order to support migration of live processes with active network connections.
5. Interface to a policy management system in order to allow dynamic, external control over communications resources used by applications.

Mockets interfaces with the agile computing kernel to provide information about connection establishment (or failure), communication statistics, and connection loss. All the network communication between the clients and the services as well as between the components of the agile computing middleware is performed via Mockets.

FlexFeed

FlexFeed provides an application-level interface that is customized for hierarchical data distribution in tactical environments. The goal is to leverage from the multi-hop nature of data distribution paths to distribute processing tasks in the path (in-stream data processing), striking a balance between data processing and data distribution.

The component relies on the agile computing framework for resource allocation and it leverages from mobile software agents to efficiently deploy filtering and fusion capabilities in the network. It differs from traditional multicast approaches because of the cyclic nature of node selection for data processing and for routing (the problem is usually solved interactively) and it differs from traditional publish-subscribe models in that data transformation components are deployed on demand (and only while needed), and there is a continuous re-evaluation of topology of the data distribution tree.

Related Publications

Further details can be found in the following publications:

Suri, N., Bradshaw, J.M., Carvalho, M., Cowin, T., Breedy, M., Groth, P., and Saavedra, R. Agile Computing: Bridging the Gap between Grid Computing and Ad-hoc Peer-to-Peer Resource Sharing. In Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003).

- Suri, N., Bradshaw, J.M., Carvalho, M., Breedy, M., Cowin, T., Saavedra, R., and Kulkarni, S. Applying Agile Computing to Support Efficient and Policy-controlled Sensor Information Feeds in the Army Future Combat Systems Environment. In Proceedings of the Collaborative Technologies Alliance Conference (CTA 2003).
- Carvalho, M. and Breedy, M. Supporting Flexible Data Feeds in Dynamic Sensor Grids Through Mobile Agents. In Proceedings of the 6th International Conference in Mobile Agents (MA 2002) Agents, Barcelona, Spain, October 2002.
- Carvalho, M., Suri, N., Arguedas, M. (2005) *Mobile Agent-based Communications Middleware for Data Streaming in the Battlefield*. In Proceedings of the 2005 IEEE Military Communications Conference (MILCOM 2005), October 2005, Atlantic City, New Jersey.
- Suri, N., Tortonesi, M., Arguedas, M., Breedy, M., Carvalho, M., Winkler, R. Mockets: A Comprehensive Application-Level Communications Library. In Proceedings of the 2005 IEEE Military Communications Conference (MILCOM 2005), October 2005, Atlantic City, New Jersey.
- Carvalho, M., Bertele, F., Suri, N., The ULM Algorithm for Centralized Coordination in FlexFee. In Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics. (WMSCI), July, 2005.
- Carvalho, M., Pechoucek, M., Suri, N. A Mobile Agent-based Middleware for Opportunistic Resource Allocation and Communications. In Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems; Defense Applications of Multi-Agent Systems (DAMAS); July, 2005.

IV. Mixed Initiative Human Control

Coordinating the activity of a team of large numbers of unmanned vehicles by a small number of human operators is a crucial part of the NAIMT research effort. However, the conventional wisdom is that this may be a very difficult, if not impossible, thing to do. Equipping unmanned vehicles with policy-based adjustable autonomy adds to the team the flexibility required for quickly shifting responsibilities in time of need, potentially easing the burden placed on the human operator. Nonetheless, complete autonomy is not a practical solution for situations involving a lot of uncertainty, or expert strategic knowledge that computers are not capable of. Therefore, it becomes a necessity to integrate humans in the coordinated activity in a natural and intuitive manner. This should involve, on one hand, an interface based on intuitive graphical displays, synergistically combined with spoken language, and on the other hand, the ability to develop, discuss, and revise collaborative plans under the framework of Human-Agent Teamwork discussed above.

As desirable as it is, a fully unconstrained dialogue system that supports true cooperative behavior is currently beyond the state of the art. For this project, we focused our research on problems with long-term benefits that also enable more limited practical systems in the short term. From a technical point of view, a key concern was developing an architecture that provides humans with intuitive and flexible control of the unmanned systems. This architecture must support:

- the ability to use contextual and linguistic constraints to enhance the recognition of spontaneous speech;
- the intuitive presentation of information with a capability to "drill-down" and present information in different ways in response to questions;
- the collaborative development of plans in which the humans and the unmanned vehicles combine their knowledge and capabilities to develop the most effective course of action in response to situations;
- the tasking and collaborative re-tasking that must occur as the situation evolves; and
- the explicit discussion and negotiation of responsibilities to define the parameters for adjustable autonomy.

Year Two Progress

During FY2003 we had completed an initial integration between our system – based on the TRIPS dialogue system for collaborative problem solving – and the KAoS framework, which enabled us to demonstrate robust and effective participation of a human operator working with a small team of real robots on a simple mine-finding task. As a consequence, our efforts in FY2004 were directed in part at incremental improvements in capabilities and robustness, with less emphasis on architecture.

Simple Logical Interface to KAoS (SLIK) – The KAoS environment provides the robotic agents with the physical communication framework as well as policy management and enforcement. The robots themselves have very limited (and potentially

heterogeneous) communication capabilities. On the other hand, TRIPS modules use much more sophisticated communication language based on speech acts. Therefore, we have developed a module (SLIK) that acts as a gateway to KAoS, keeping the undesired low level communication hidden from both TRIPS and the human user. SLIK translates collaborative problem-solving acts from TRIPS into commands for the robotic agents, and wraps the answers from the agents into KQML messages. By using SLIK, the rest of TRIPS is protected from the limitations and different semantics of robot agents and can continue processing as if every agent has the same communication abilities. As the agents progress in sophistication, SLIK is the only TRIPS component that needs to change while its interface will always appear the same.

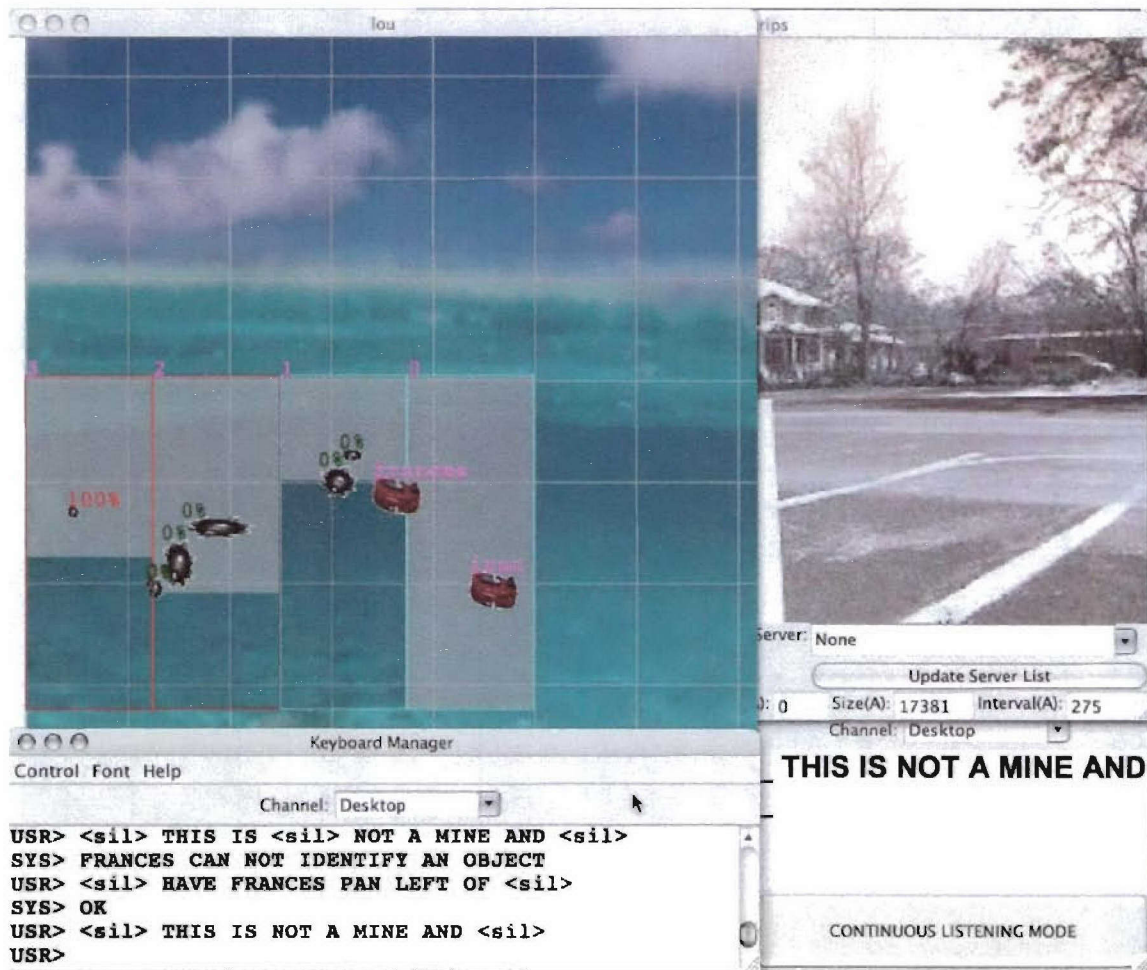
Ontology Mapping – In Year One we developed an initial mechanism for mapping between the KAoS ontology and the TRIPS ontology. This year we developed that mechanism into a general module for transforming the semantic, domain-independent logical form (LF) that is produced by the TRIPS Parser into any domain-specific knowledge representation (KR), not just the KAoS ontology. The algorithm uses a database of hand coded mapping rules that match patterns in an LF and generates the appropriate KR. Specific rules are used to directly create a corresponding KR concept, while general rules are used to handle more general constructions (such as the agent of an Action). The resulting KR is used by domain specific reasoners and also to communicate with external components. New mapping rules are created to map the LF into the KR for each domain. The advantage of this approach is that these rules are easier to create than a new logical representation for every new domain.

Multi-Modal Displays – The graphical interface is of paramount importance for helping the user quickly assess the situation. At the same time, great economy of expression can be obtained by using active displays in conjunction with speech (for example, by selecting an area of the map and saying “Search *this* area.”). Building on the success of this model in Year One, we decided to re-implement the graphical interface to support additional situation awareness and active input/output capabilities. The new active map allows the user to zoom in and out, focus on particular areas, show and hide objects to increase display clarity, etc. In addition, the map shows at-a-glance information about the situation, such as:

- Status of a lane (cleared, mined, unknown);
- Names for robots and lanes, so the user can refer to them by name;
- Robots’ assessment of the likelihood that an object is a mine, so the user can prioritize actions according to the situation;
- Likely size of the objects, using proportional rendering, when such information is available from the robots.

This year we also integrated a video display that obtains video from the robots via FlexFeed. The user can ask the robots to provide video at different resolution levels, and also to zoom in and out, pan left and right, etc., so the user can assist in identifying objects.

Spoken Language Recognition and Understanding – Much of the work in this area has been incremental. Building on the Year One system, we have made improvements to the



TRIPS user interface. Shown are the active map, the dialogue recent history, the speech recognition result for the current utterance, and a portion of the video display.

lexicon, language models, and the ontology to insure that the system has the knowledge to understand the user's utterances as the complexity of the scenario (and hence of the language) increases.

Resulting Papers

- N. Chambers, J. Allen, L. Galescu, and H. Jung (2005). A Dialogue-Based Approach to Multi-Robot Team Control. In *Proceedings 3rd International Multi-Robot Systems Workshop*. Washington, DC.
- N. Chambers (2005). Real-Time Stochastic Language Generation for Dialogue Systems. In *Proceedings European Workshop for Natural Language Generation*, Aberdeen, Scotland.

References

- Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2000). An Architecture for a Generic Dialogue Shell. *Journal of Natural Language Engineering*, 6(3), 1-16.
- Allen, J. F., & Perrault, C. R. (1980). Analyzing Intention in Utterances. *Artificial Intelligence*, 15(3).
- Chambers, N., & Allen, J. (2004). Stochastic Language Generation in a Dialogue System: Toward a Domain Independent Generator. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, Boston, USA.
- Cohen, P. R., & H. J. Levesque (1980). Speech Acts and the Recognition of Shared Plans. Paper presented at the *3rd Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, Victoria, B. C.
- Ferguson, G., & Allen, J. (1998). TRIPS: An Integrated Intelligent Problem-Solving Assistant. Paper presented at the *NCAI (AAAI-98)*, Madison, WI.
- Galescu, L., Ringger, E. & Allen, J. (1998) Rapid Language Model Development for New Task Domains. *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, Granada, Spain.

FIHMC Travel Authorization Request (TAR)

Number:

Date of Request: 02/06/06		Blanket TAR <input type="checkbox"/>		FIHMC	
SHEPPARD, Julie		018-38-7729		FIHMC	
Traveler's Last name, First, and Initial		Social Security # /		Official Headquarters	
<input checked="" type="checkbox"/> Employee <input type="checkbox"/> Non-employee <input type="checkbox"/> Student					
PURPOSE OF TRIP: Legislative meetings.					
Others going on same trip:					
Is expenditure reimbursable from other source(s)? YES <input type="checkbox"/> NO <input checked="" type="checkbox"/> Source(s):					
Pre-payment of registration fee? YES <input type="checkbox"/> NO <input type="checkbox"/> Amount: \$		Deadline Date:		FEID #	
6:00am 02/12/06		Washington, DC		11:00am 02/14/06	
HOUR and DATE of departure		DESTINATION(S)		HOUR and DATE of return	
Statement(s) of benefits to the State: (Required for all Conventions, Conferences, Workshops, etc.)					
Other Accounts to be Charged: [1] \$ [2] \$ [3] \$					
Pursuant to Section 112.016(3)(a) F.S., I hereby certify or affirm that this travel is on official business of the State of Florida and will be performed for the purpose(s) stated:					
Traveler's signature		Date:			
Supervisor's signature		Date:			
(Supervisor must have FINAL official budgetary authority. Other may enter their initials only.)					
Dean/Director's signature		Date:			
Vice President's signature		Date:			
President's signature		Date:			
(President's signature required for all travel in excess of 30 days)					

ESTIMATED COST	
AIRFARE / COMMON CARRIER	Travel Agency
Travel Agent	Phone No.
To be paid by:	
<input type="checkbox"/> University Central Billing	PCard
<input checked="" type="checkbox"/> Traveler (Reimbursed after travel)	
Day(s) per diem @ \$50.00/day	\$ 269.00
Day(s) single room rate @ \$ /day	\$ 269.00
Hotel	PCard
Meals: (Pier diem based on Destination)	\$ 176.00
Miles @ 36 cents per mile	\$
Incidental Expenses (receipts required)	\$ 75.00
Registration Fees	PCard
Rental Car	Yes <input type="checkbox"/> No <input type="checkbox"/> Conf #
1 TRIP TVL VCHR	YES <input type="checkbox"/> NO <input type="checkbox"/> PCard
TOTAL ESTIMATED EXPENSES	
	\$ 789.00

JUSTIFICATIONS/SPECIAL INSTRUCTIONS: